UNIVERSIDAD CARLOS III DE MADRID

ESCUELA POLITÉCNICA SUPERIOR

BACHELOR IN TELEMATICS ENGINEERING


BACHELOR THESIS

# GENGHIS, AN AUTHORING TOOL FOR KHAN ACADEMY EXERCISES

Author: Juan Luis Sanz Moreno
Supervisors: Pedro J. Muñoz Merino
Carlos Delgado Kloos

September 24, 2014

# Dedication

This degree has been an arduous and exciting journey that in the five years that have passed since its beginning, has brought many changes to my life, personality and way of perceiving the world that surrounds me. These changes have not been the sole work of study and learning, but of the people that I have had the tremendous pleasure of sharing these years with, and for that, I am truly thankful to you all, but I would like to spend a few lines on some of the people that have helped me the most in what has been the journey if a lifetime.

First, I would like to thank my parents, Juan Luis and Africa, together with my sister Africa for always supporting me, no matter the adversity. Thank you for helping me make my choices and believing that I could, not only do this degree, but do a great job at it.

Secondly, I would like to thank my girlfriend Patricia, who has lived with me the experience of studying a degree at UC3M and has always been there for me, has always known when I needed help, support or encouragement and without whom I would have not made it to where I am today.

An enormous thanks to my supervisor and mentor along this project: Pedro J. Muñoz Merino, who has tirelessly helped me with knowledge and advice in any issue that I have ever had. To you Pedro, thanks for everything you have done for me.

I need to also thank my friends and classmates at UC3M: my friends at the Telematics degree Alvaro, Lavin, Nicolas, Manu and Zenitram, for helping me become a better engineer and create the best learning environment a person could ask for. Thanks to you and the rest of the people that have accompanied in various subjects along the way: Luis, Carmen, Estefania, Jorge, Rocio, Monica, Yaiza, Pilar, Teresa, Alex, Dani and Javi. Thank you all.

A very special thank has to go to my colleagues at Eleven Paths, to my team and bosses Ioseba, Manu, Alessandro, Pablo, Chema and a very personal thanks to Cristóbal, Julia and Dani, who have been, without a doubt some of my biggest supporters in the latest year of my degree and with this project. To Cristóbal I owe not only unshakable support, but the subject you helped me with. To Dani, whom I own a lot of what I have learnt in the last months. Finally, to Julia, whom I owe the support, help and advice she has given me.

Finally, and most certainly not least, to the professors at UC3M that have helped me through my degree and have taught me more than knowledge. Thank you Julio Villena, Carlos Delgado, Raquel Crespo, Aberlardo Pardo, Emilio Parrado, Carlos J. Bernardos, Carlos G. Rubio, María Celeste Campo, Maria Blanca Ibañez, Antonio de la Oliva, Antonio Serrano, Manuel Urueña, Ivan Vidal, Fernando P. Cruz, Jeronimo Arenas, David Ricardo Sanchez and Luis Fernando de Inclan.

# Abstract

With the rapidly growing use of the Internet, the use of Massive Open Online Courses (MOOCs) is spreading through the academic community, in light of the benefits these provide not only to the students but to society as well.

Most of these courses generally contain some form of evaluation on which a student must prove, either to himself or for evaluation purposes, the learning he has acquired along the way, generally in the form of exercises assigned to a specific topic or area of knowledge and have the characteristic that, most of the time, do not require human supervision or correction.

The main objective of this project is to create an authoring tool for professors without any technical background to generate the aforementioned exercises in a simple, elegant way with the possibility to preview the exercise in progress and formula-editing tools for easy, simple calculations.

Moreover, this tool is to be tested on a real production environment with real end users who would test not only the authoring tool, but experience the first implementation of these courses at UC3M.

**Keywords**: MOOC, Khan Academy, Authoring tool, MVC.

# Contents

**1 Introduction** **13**

1.1 Motivation . . . . . 14

1.2 Objectives . . . . . 15

**2 State of the art** **17**

2.1 Khan Academy . . . . . 17

2.2 Other MOOC platforms . . . . . 20

2.2.1 Coursera . . . . . 20

2.2.2 edX . . . . . 21

2.3 Alternative authoring tools . . . . . 21

2.3.1 eXe . . . . . 21

2.3.2 Adobe Captivate . . . . . 22

2.3.3 CourseLab . . . . . 23

2.4 Other markups and standards for online exercises . . . . . 24

2.4.1 SCORM . . . . . 24

2.4.2 IMS Question & Test Interoperability . . . . . 24

**3 Design** **27**

3.1 Initial brainstorm . . . . . 27

3.2 Post-brainstorm investigation . . . . . 29

3.2.1 Authentication possibilities . . . . . 30

3.2.2 Management view . . . 31
3.2.3 Data persistence and consistency . . . 31
3.2.4 Server-side . . . 33
3.2.5 Client-side . . . 37

**4 Development** **41**

4.1 Early stages . . . 41
4.2 Databases . . . 43
4.3 View scaffolding . . . 45
4.4 Controllers . . . 49
4.4.1 Variable Controller . . . 49
4.4.2 Hint controller . . . 49
4.4.3 Parser . . . 49
4.5 Front-end . . . 54
4.5.1 JavaScript for variable deletion . . . 54
4.5.2 TinyMCE . . . 55
4.6 Rework of the view . . . 56

**5 Deployment and evaluation** **59**

5.1 Quality Assurance . . . 59
5.2 Deployment . . . 60
5.3 Presentation and evaluation . . . 61

**6 Conclusions and Future work** **65**

6.1 Conclusions . . . 65
6.2 Future work . . . 67

**A Git deployment** **71**

**B Model mapping** **73**

B.1 The variable model . . . . . 73

B.2 The exercise model . . . . . 73

B.3 The user model . . . . . 74

# List of Figures


2.1 Early design of Khan Academy . . . . . 18

2.2 September 2014 version of Khan Academy . . . . . 19

2.3 A sample Coursera video lesson . . . . . 20

2.4 eXe interface . . . . . 22

2.5 Adobe Captivate 8's authoring tool . . . . . 23

2.6 QML example[25] . . . . . 25

3.1 Initial design . . . . . 29

3.2 Login design . . . . . 30

3.3 Initial management design . . . . . 31

3.4 The model - view - controller schematic[15] . . . . . 34

3.5 Sample jQueryUI tabs . . . . . 39

3.6 Sample Bootstrap modal . . . . . 39

3.7 jQueryUI Draggable elements . . . . . 40

4.1 Machine schematic . . . . . 42

4.2 Machine Schematics with Databases . . . . . 44

4.3 Database Schematics . . . . . 46

4.4 An example of how Bootstrap's column system works . . . . . 48

4.5 First design of the edit exercise View . . . . . 48

4.6 Adding a variable . . . . . 50

4.7 Deleting a variable . . . . . 51
4.8 Editing a variable . . . . . 52
4.9 TinyMCE custom configuration . . . . . 56
4.10 TinyMCE LaTeX formula insertion table . . . . . 56
4.11 TinyMCE custom variable dropdown . . . . . 57
4.12 Final design . . . . . 57
4.13 Preview tab of the final design with real exercise . . . . . 58

5.1 Inclined plane exercise image . . . . . 62

# Chapter 1

# Introduction

Over the last decades, the growth and widespread availability of the Internet has brought about a new wave of changes for all sectors of society and while some may have gotten greater benefits from these changes, it can be agreed that the increase in globalization and spread of ideas has transformed the way many elements in our everyday life work.

The economy, health, trading, communication or even the food industry have been adapting to these new technologies and opportunities as they were presented, some in very similar aspects, other in very unique approaches.

There was, however, a part of our culture that has remained untouched for many decades: the learning and teaching process. Our great great grandfather were taught sitting in desks, in front of a blackboard by writing down notes on a paper a professor with a greater deal of knowledge would present to them, and not much has changed since, at least not until fairly recently.

Ten years ago, back in 2004, 14 out of the 43 million Spaniards had Internet access at home [3] and although the percentage is low, its grown to this date has amounted to a 62% of Spains population. With all these people connected to the Internet and, by extension, to the entire world, could no change be done to further enhance education and the way it has been done for the last century?

Many engineers, tech users and entrepreneurs tried to open the world of education to the Internet, some with more success than others, but it was not until a certain event that this sector would see the change it is undergoing right now, which will most likely change forever the concept of education: MOOCs

MOOCs or Massive open online courses are a fairly recent approach to education, in which any user can sign up for a course that has been split into sections, each explained in its entirety on the Internet, with no traditional classes (most of them) and while it has sometimes been labeled as a competitor to the more classical approach to teaching like universities, this threat may not be as significant as it was initially predicted[2].

These courses can, at least a great percentage of them, be seen as a gathering of a lot of different elements among which two are part of the core of the experience: the teaching of the

subject and the exercises.

The widespread of video content on the Internet, thanks to sites such as youtube.com [4] which enabled everyone to, in easy steps, make a video about any subject he or she desired and share it for free with as many other users as he or she pleased, and this opportunity was also adopted by many in the learning industry, but mostly pioneered by Salman Khan, founder of Khan Academy.

Khan Academy[5] started back in 2006 with Salman Khan as its only member and has since grown at gigantic steps, surpassing any other educational YouTube channel in views and thousands of users every day learning and taking courses on any subject available.

It was the other part of the equation, the exercises, that would come later to maturity. Khan Academy released in mid-2011 an open source engine, capable of transforming an HTML file with a specific set of rules into a fully functional exercise, in which each student would get the chance to exercise not only on simple arithmetic, but with the help of its powerful engine, graphing calculus geometry and a wide range of exercise types, each with a different set of values for each student, making each exercise unique[6]. These exercises in question were sometimes a little too complex to be understood by professors with no programming background, as learning the basics and use of not only HTML, but JavaScript as well, would prove sometimes too troublesome for the potential rewards, that is when Genghis' authoring tool would prove useful in aiding professors in this effort.

The idea behind the Genghis authoring tool was to create a simple application that professors, or any user without technical knowledge could use to make Khan Academy styled exercises, to use in their open source engine.

## 1.1 Motivation

There are several factors that have contributed to the motivation behind the project of Genghis and its authoring tool. It was however, primarily pushed and thought of by University Carlos III's vice-dean in infrastructure, Carlos Delgado Kloos. He approached several professors with the idea of making a MOOC platform for beginner students at UC3M, on which they could find all the necessary material required to successfully take their degrees from the beginning in elementary subjects, such as math, physics or chemistry.

These courses already existed in a more classical way, students would come in a few weeks early their first year to review and ensure they have the required level of knowledge to start their courses with the right background knowledge so as to not feel lost from day one. UC3M's vice-dean's idea was to use these courses as an early version of what could become the university's MOOC platform by complimenting these classes with online lessons and exercises to be taken as early as August to begin with a solid foundation [7].

The idea was received with a warm welcome and the plan started to form more clearly: the university will set up a copy of Khan Academy's MOOC platform and professors would be assigned their respective course, in which they could upload short, 10 minute videos explaining parts of different topics. There was, however, a problem.

Part of the MOOC experience is, as mentioned before, a form of evaluating the knowledge gained thought the experience, be it from videos or any other source. This form of evaluation existed but in the form of exercises that had to be programmed beforehand, field in which some professors were not very comfortable with.

This is when the vice-dean approached what is now the development team with the task of creating a tool to help these professors make those exercises for a complete MOOC experience for their students.

Shortly after introducing the core ideas and workflow, the creation of Genghis began.

## 1.2 Objectives

This section is dedicated to a brief overview on Genghis' authoring tool's objectives. These will, however, be explained in more detail in section 3 and 4.

The main objective of the authoring tool is, as mentioned before, to create a simple application in which professors can create Khan Academy styled exercises in the form of fill-in-the-blank questions. These exercises would then be passed automatically to the Khan Academy exercise engine, where they will be transformed into complete HTML exercises.

The other main goal of Genghis is to assist professors in preparing the test MOOC that was to take place in mid-August, where new freshmen would join UC3M in the beginning courses (a.k.a. "Cursos cero") where they would learn and review the basic knowledge required to successfully start their degrees.

These were, however, not the only goals at hand, but rather accompanied by the following:

- The tool must be simple.
- The design must be clean and efficient.
- The user experience as a whole has to be satisfactory.
- The process of creating an exercise must be straight forward.
- Genghis must be finished in time for its final purpose: to be used by professors to create exercises for new, upcoming MOOCs.
- The tool must be integrated in the GEL platform, from which professors would access to their exercises and data.
- The tool must be hosted using version control to avoid code being lost and easing the deployment phase.

# Chapter 2

# State of the art

The purpose of this chapter is to provide the state of the art for Genghis, Khan Academy and authoring tools, as well as other MOOC platforms other thank Khan Academy. Section 2.1 will provide background information on Khan Academy, platform for which Genghis' authoring tool creates exercises. Sections 2.2 and 2.3 will present other alternatives to Khan Academy and authoring tools. Finally, section 2.4 will be focused on the presentation and discussion of alternative markups for exercises online.

## 2.1 Khan Academy

To better understand the landscape of massively open online courses, an explanation of Khan Academy is due since it was, after all, one its precursors and early testers of this new approach to education.

Khan Academy had a small and rather peculiar beginning, Salman Khan, its founder and three times graduate from MIT and holder of an MBA from Harvard, was tutoring his cousin when more family members wanted to jump in on the tutoring. Mr. Khan, overloaded with tutoring demands, decided to try something that had not been tried to that extent before, to record his lessons in short YouTube videos using a simple drawing program to screencast the exercise or lesson's progress as he advanced in its explication[9]. This was very well received and the popularity of these videos grew until Mr. Khan decided to open his own site: Khan Academy, on which he'll sort the links to his YouTube videos by subject. An early 2009 version of Khan Academy can be found in figure 2.1.

From then on, Khan Academy exploded in popularity. Mr. Khan dropped his job at a hedge fund to work full time in Khan Academy, hiring in the process a full staff to help him with not only the videos, but exercises and the web site as well which as of September 2014, includes 73 people employed and a dog, who is also considered part of the team[8]. There is also a great amount of volunteers who have contributed to the translation of Khan Academy's website to 23 languages and its videos to over 60 different languages.

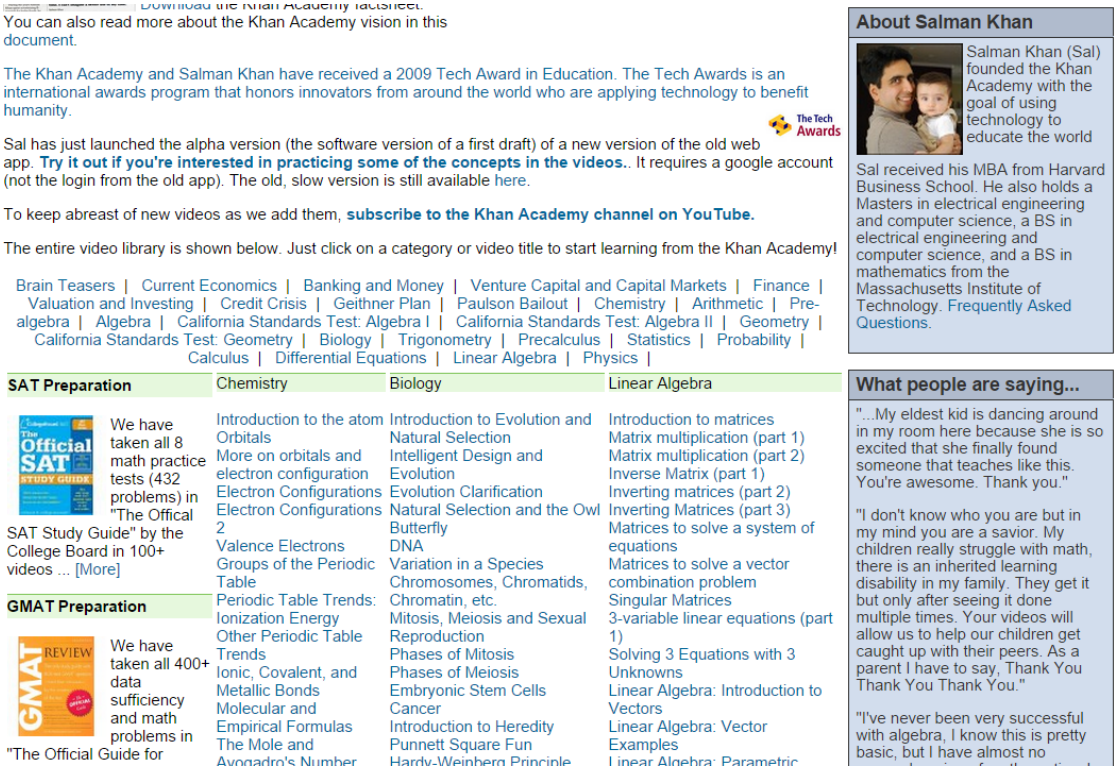


Figure 2.1: Early design of Khan Academy

Since the early versions like the one depicted in figure 2.1, Khan Academy has changed not only its appearance, but the organization as well. Videos are no longer linked on the main site, but rather organized in courses, with tree-like structures which to follow for the different topics each course may have (refer to figure 2.2 for a September 2014 version of Khan Academy). There has also been included a social factor, in which users can share the badges they have obtained through gamification by means of different feats, be it finish a certain percentage of all the videos of a course, complete entire subjects and all of those badges related to the completion of exercises.

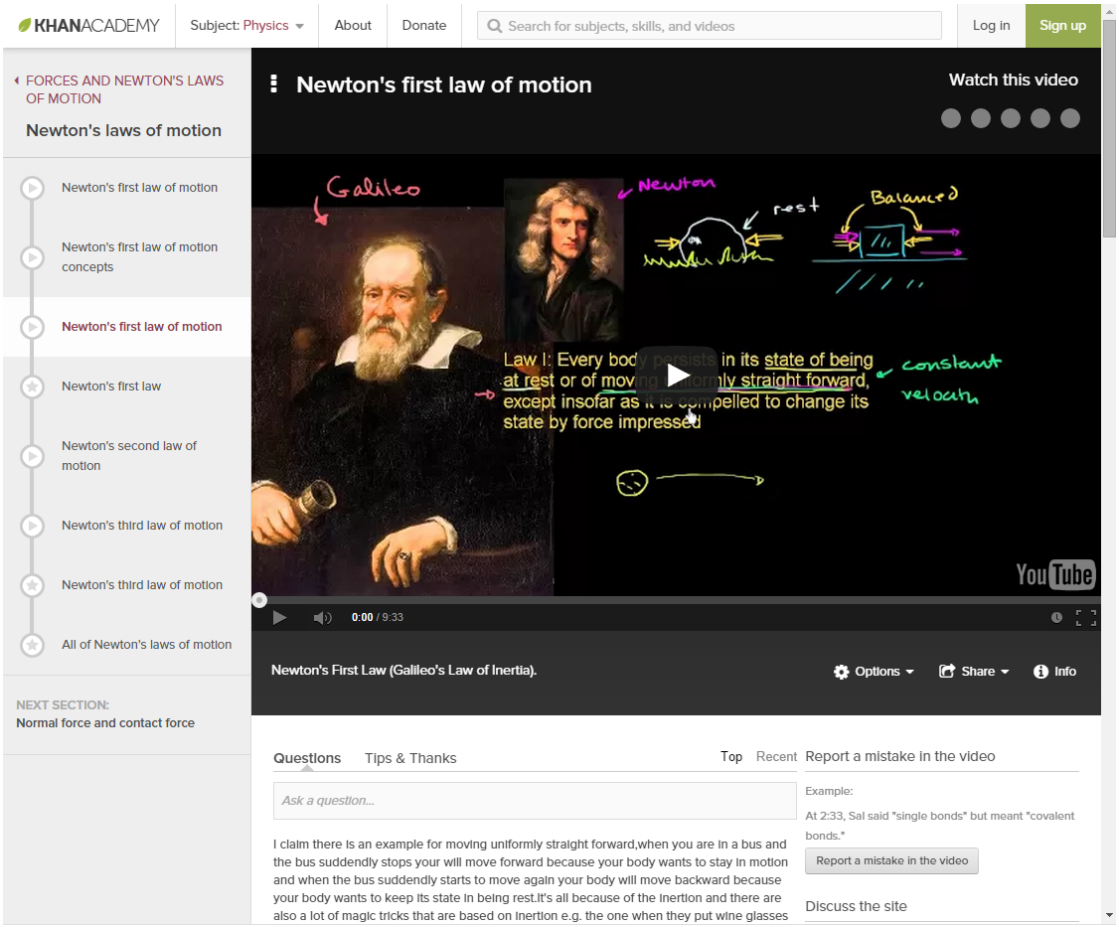


Figure 2.2: September 2014 version of Khan Academy

The exercise module was released in mid-2011 and was composed of a set of files that would process a specific HTML structure in order to form a complete Khan Academy exercise. These exercises had the characteristic of being dynamic, as the variables used in the exercises could be set to randomize over an interval, rendering a different solution each time a student access the exercise[6]. It was this characteristics and the capability to offer hints to exercises at the student's request that made this module an extremely versatile one and the reason it was chosen at UC3M for the MOOC courses.

A sample of the HTML code parsed by Khan Academy's exercise engine is shown in page 53.

## 2.2 Other MOOC platforms

Of course, Khan Academy was not alone in the MOOC business. The interest in providing education to everyone quickly spread across academic entities and private companies, seeing the social and potentially economic benefits it could offer, and many of these developed their own MOOC platforms, with varying results in terms of success.

Of the hundreds of possible alternatives, 2 will be discussed in this section: Coursera[10] and edX[11]

### 2.2.1 Coursera

Designed and Developed by two professors at Stanford University, Coursera is a MOOC platform open to universities to make their courses online and available on the Internet, with 9.2 million users as of September 2014, it is one of the biggest MOOC platforms in the world and unlike Khan Academy, Coursera is not Open Source and cannot be installed on a personal server for private use.

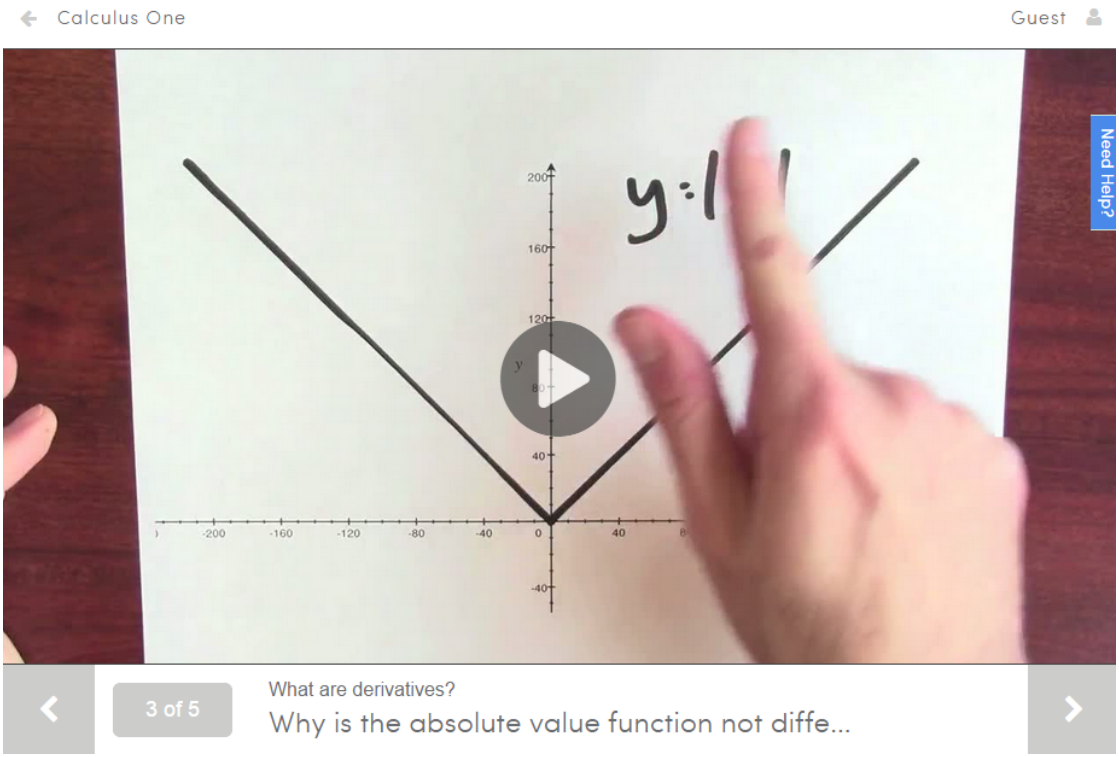


Figure 2.3: A sample Coursera video lesson

Coursera also added recently the possibility to add exercises to subjects, not only exams, but it does not have the possibility to help the student with hints not is it as versatile as Khan Academy's.

### 2.2.2 edX

edX is a completely free and open source MOOC platform, created by Harvard University and the Massachusetts Institute of Technology in mid-2012 to offer a very wide array of subjects and courses by an approved set of universities to their users.

This MOOC platform has recently seen a growth in their user base, which has now reached over 3 million users[12], all thanks to their high quality, free courses and the recognition it offers for a small fee (even if the course is free).

edX was on a very early stage at the beginning of the Genghis project, but they do have as of September 2014 an exercise module to render exercise and check the solution, but again without as many features as the ones Khan Academy offers and even offered back on edX's release date.

## 2.3 Alternative authoring tools

Of course, the authoring tool included in Genghis was not the first one created with this purpose. Several tools have been created with the purpose of helping people with little programming background create exercises, be it for a later processing in an external parser like the Genghis authoring tool, whose output was to be parsed by Khan Academy's exercise module or directly exported to HTML.

Out of the several tools that can be found with this purpose, a selection of three of the most used tools will be presented in order to set a background on which to compare the Genghis authoring tool against.

### 2.3.1 eXe

eXe is an open source project initially developed in New Zeland by the University of Auckland, The Auckland University of Technology, and Tairawhiti Polytechnic. It is a downloadable tool with the necessary elements to create several forms of exercises like multiple choice, fill in the blank or multiple select and export these exercises to several formats.

eXe exercises are created following a series of steps, listed below.

1. **The Outline**: for the creation of exercises, the user must first input the tree-like structure that will define these exercises by creating a hierarchy of the elements that compose it, for example course, outline, objectives and the exercises themselves. (Top left corner of figure 2.4)
2. **iDevices**: iDevices, or instructional devices, are the core elements of the exercises. These are templates for the several parts that can the exercise block can contain, such as exercise types, images, case studies or external websites. (Bottom left corner of figure 2.4)
3. **The authoring**: The authoring is the actual fill of the iDevice that was added in the previous step. Once an iDevice is chosen, the user can edit the template to his needs.

(Right section of figure 2.4)

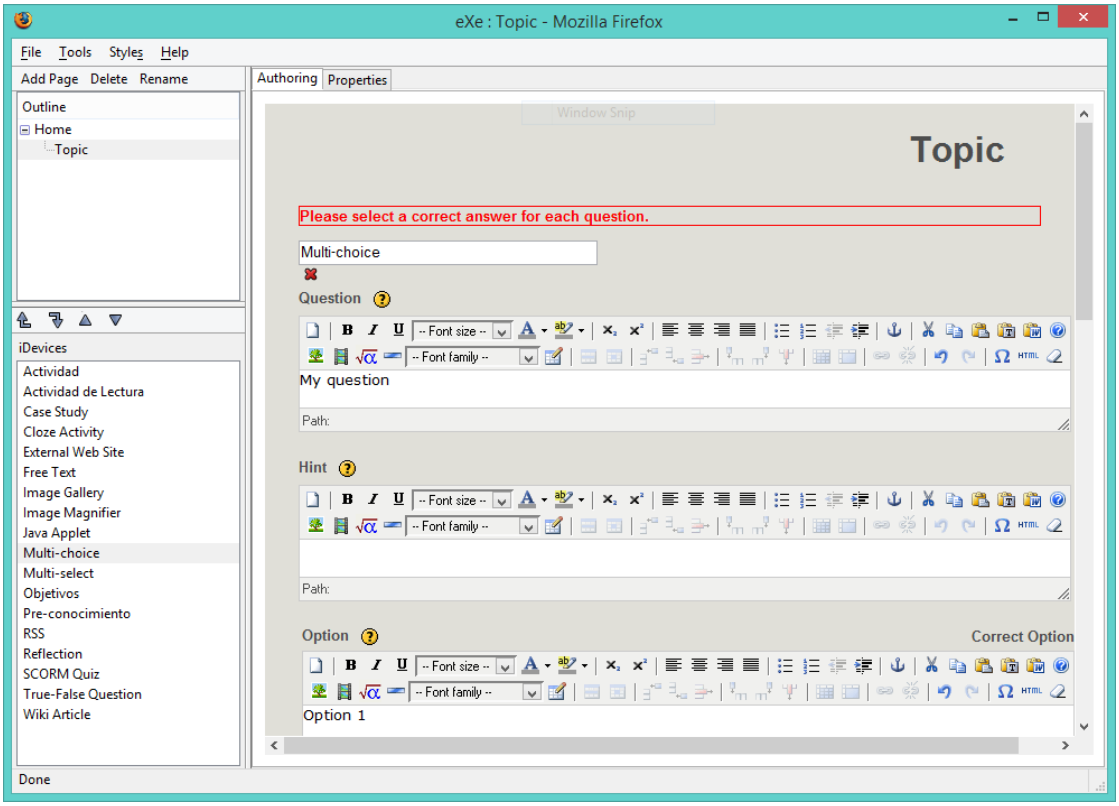


Figure 2.4: eXe interface

### 2.3.2 Adobe Captivate

Adobe's latest tool to create multi-device, responsive e-learning tools and exercises. This non-free tool is oriented to the creation of presentations. Its authoring tool allows the user to create a series of slides on which he can select what is known as a "quiz slide" on which he can add question in several formats, namely:

- Multiple Choice
- True/False
- Fill-In-The-Blank
- Short answer
- Matching
- Hot Spot

- Sequence
- Rating scale
- Random question

Together with the amount of these that the user would like to add. Then the purpose of the exercises (grading, survey or pretest)

Once selected, the user can edit each exercise previously inserted to his own needs. The editor is shown in figure 2.5

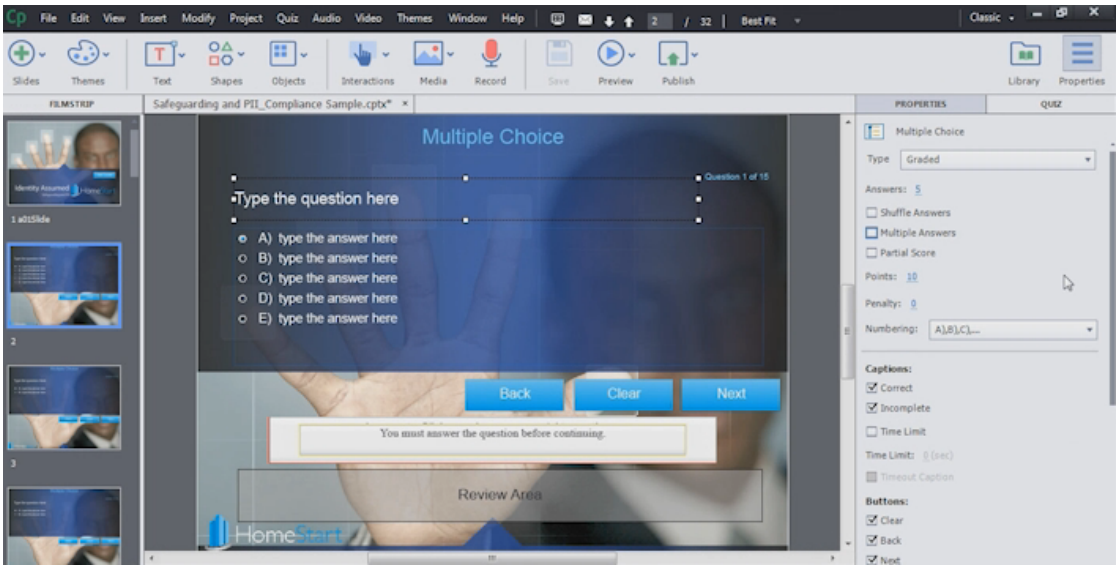


Figure 2.5: Adobe Captivate 8's authoring tool

### 2.3.3 CourseLab

CourseLab, much like Adobe Captivate, is a tool focused on making slide-like presentations and exercises. The interface is very similar to the latest versions of Microsoft Office Power Point and offers a variety of options and possibilities, ranging from Flash, Shockwave and Java Applet embedding to the creation of various forms of assessment, including:

- Single choice
- Multiple choice
- Ordered items
- Numerical fill-in-the-blank
- text fill-in-the-blank

- Matching pairs

It offers a WYSIWYG editor for easy customization, scoring mechanisms and exporting to various standards including HTML, AICC and SCORM.

## 2.4 Other markups and standards for online exercises

As it was mentioned before, Khan Academy uses a specific markup for their exercises that is fed to the engine to produce a fully complete HTML exercise in Khan Academy style. This, however is not the only nor the first standard for learning management systems (LMS). The two most used are presented below:

### 2.4.1 SCORM

SCORM, or Sharable Content Object Reference Model, is a set of specifications and standards with purpose of making all e-learning content equally valid across different LMS.

SCORM's official website[24], apart from other information on the standard and its engine, offers a very interesting analogy:

> Lets take DVDs for example. When you buy a new movie on DVD you dont need to check to see if it works with your brand of DVD player. A regular DVD will play on a Toshiba the same as it will on a Panasonic. **Thats because DVD movies are produced using a set of standards**.

The idea behind SCORM is making all the content generated by any of the authoring tools explained in section 2.3 and any other tool with similar purpose interchangeable and homogeneous.

It is very important to note that Khan Academy does not follow the SCORM standards, so any exercise developed with the Genghis authoring tool cannot be used on other tools specifically adapted to do so.

### 2.4.2 IMS Question & Test Interoperability

Like SCORM, IMS QTI is a set of specifications designed with the purpose of creating standard for all tests and result data from authoring tools and LMS to be authored interchangeably from one system to another.

It started by using a proprietary markup language known as QML (Questions Markup Language) which has changed and evolved into a markup that can now be used for any form of question.

The standard is set in a non-HTML markup language, hence it resembles very little similarities with those exercises of Khan Academy. A sample of the multiple choice exercise shown in figure 2.6 is written below.

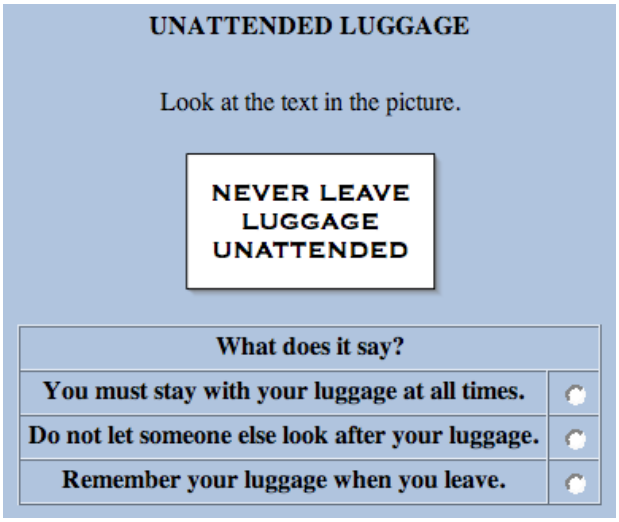


Figure 2.6: QML example[25]

```
<!--
 The example adapted from the PET Handbook, copyright University of Cambridge ESOL Examinations
-->
<assessmentItem
    xmlns="http://www.imsglobal.org/xsd/imsqti_v2p1"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="http://www.imsglobal.org/xsd/imsqti_v2p1 http://www.imsglobal.org/xsd/qti/qtiv2p1/imsqti_v2p1.xsd"
    identifier="choice"
    title="Unattended Luggage"
    adaptive="false"
    timeDependent="false">
    <responseDeclaration identifier="RESPONSE" cardinality="single" baseType="identifier">
        <correctResponse>
            <value>ChoiceA</value>
        </correctResponse>
    </responseDeclaration>
    <outcomeDeclaration identifier="SCORE" cardinality="single" baseType="float">
        <defaultValue>
            <value>0</value>
        </defaultValue>
    </outcomeDeclaration>
    <itemBody>
        <p>Look at the text in the picture.</p>
        <p>
            <img src="images/sign.png" alt="NEVER LEAVE LUGGAGE UNATTENDED"/>
        </p>
        <choiceInteraction responseIdentifier="RESPONSE" shuffle="false" maxChoices="1">
            <prompt>What does it say?</prompt>
            <simpleChoice identifier="ChoiceA">
                You must stay with your luggage at all times.
            </simpleChoice>
            <simpleChoice identifier="ChoiceB">
                Do not let someone else look after your luggage.
            </simpleChoice>
            <simpleChoice identifier="ChoiceC">
                Remember your luggage when you leave.
            </simpleChoice>
        </choiceInteraction>
```

```
    </itemBody>
    <responseProcessing template="http://www.imsglobal.org/question/qti_v2p1/rptemplates/match_correct"/>
</assessmentItem>
```

Another interesting feature of the IMS-QTI engine is that it grants the possibility to have adaptive exercises, for example allowing the question to adapt to the level of knowledge of the student in particular[1].

# Chapter 3

# Design

The purpose of this chapter is to develop and explain the main decisions taken before the implementation of the tool, including the front-end and back-end of the tool, the feedback from the original brainstorm and the final decisions taken before moving on to the creation and development of the authoring tool.

## 3.1 Initial brainstorm

As with any medium to large project, brainstorming is a crucial first step often overlooked, since a good brainstorm, combined with a good initial feedback provides an ideal, solid ground on which to build any application.

For our authoring tool, the original idea was straight-forward: to create a simple, intuitive authoring tool to create Khan Academy styled exercises so professors could create the aforementioned exercises themselves without requiring any technical knowledge and use them wherever they pleased. This, however, turned out to be a far simpler thing being said than done, as with any project, a great deal of issues had to be tackled and dealt with before any development could start, since making a tool so simple as it was planned to be, would not an easy task.

To achieve the rather complex goals above, it was decided that the following were absolutely necessary features:

- As many page elements as possible should be grouped on the same view, without cluttering the page excessively.
- A way for users to sort and organize their exercises.
- Some form of exercise preview.
- A way to export the finished exercises.

As to why these features were deemed so important:

The main reason for having as many of the page elements as possible in the same view was that we wanted the users to have the edition of an entire exercise in the same space an exercise is: a single view. In fact the user interface development eventually lead to a structure similar in order to a finished exercise, giving the user a more intuitive approach to the tool, since it flows the same way the finished product does.

Having a way to organize the exercises was conceived with the following main features in mind: to have the user be able to see all their exercises, differencing which is which by their titles and possibly a small extract from the exercise's statement, users may also be given the option in this view to delete any exercise from the list.

Thirdly, the initial brainstorm also concluded that there should be some form of preview for the exercise being currently edited, be it in a separate tab, a pinned area in the exercise view or a section of the view that would slide over part of the exercise (or the entire view). Either way, the exercise had to have a way of being previewed, as professors or any end user can make a mistake (in the formulas specially, as any non-closing parentheses can change the entire meaning of an equation) and having a quick way to preview the exercise could potentially reduce the number of these mistakes that make it to the students.

It was also deemed necessary to include a way for users to export their work. Since the decision of whether to develop this tool as a Moodle plug-in or a separate web application had not been made at the time until further research on the pros and cons, it was decided that either way, there should be a way for users to save their finished exercises in their machine, as well as a way to send it directly to the final production server where students could start working on the exercise (publishing).

There were also some other aspects discussed on this meeting, such as the importance of a fast interface for the experience to be as enjoyable as possible, the use of as many components as possible that were not only simple, but also familiar to the ones professors use on a daily basis, such as the editor, and lastly whether it would be possible to add LDAP as a means of log in (so professors would not have to create separate accounts).

Finally, it was time to draw a small mockup of what the main structure of the interface would look like, a wire-frame copy of it is shown in figure 3.1 on page 29.

In this figure, we can observe most of the aspects that have been discussed in the previous paragraphs, specifically, that the entire exercise fits in one single view, that the flow of the exercise is the same as the order in which the sections of a full exercise are presented, enhancing user experience and adding an intuitive form of navigation to the site. It is also worth noting that the view is split in two, showing in the right hand side of the screen a preview pane, where the user would be able to see his/her progress on the exercise without having to open separate windows. It was taken into consideration the possibility to include tabs in the preview pane, to allow the user to see the HTML markup that was being generated as an alternative (or additional) way of exporting the exercise.

The disposition of the view that will let users see and organize their exercises was, although discussed, not included in these mockups, since it was decided that the main focus for now should be on the exercise itself, and not on the management view or the login options. These however had to be taken into consideration later, in the investigation phase that was to happen before the feedback.

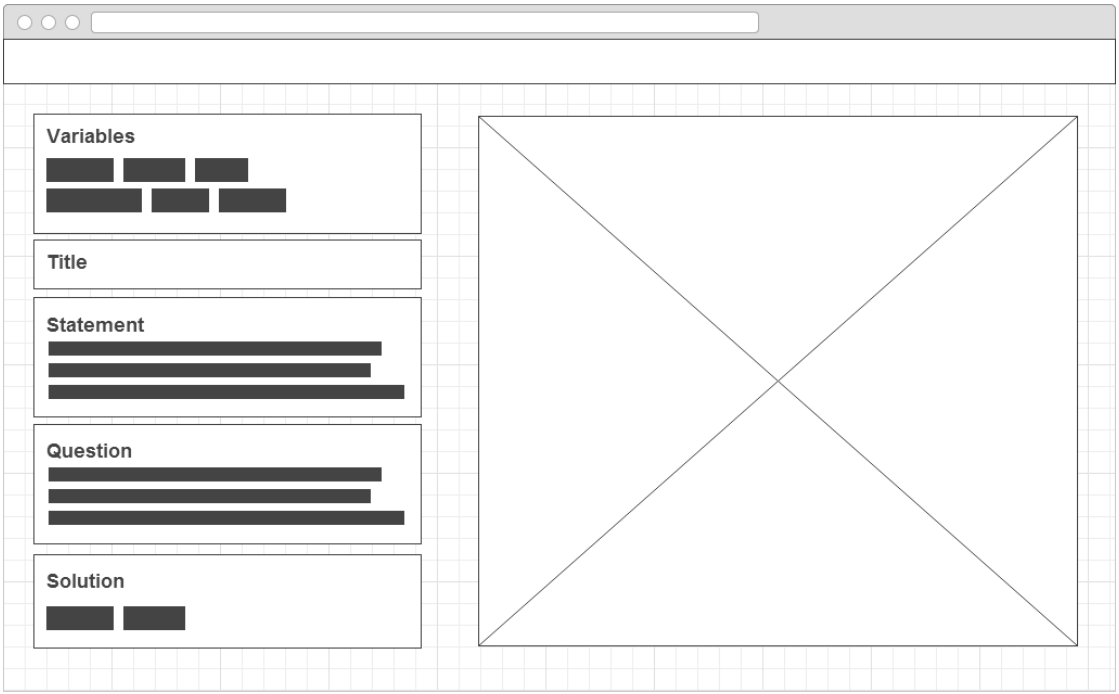


Figure 3.1: Initial design

## 3.2 Post-brainstorm investigation

The issues mentioned before in section 3.1 were deemed important from the very beginning, making them a requirement for the project. However, there were some other less critical decisions that had to be made, but these required a further research, these were:

- Authentication possibilities.
- Management view.
- Environment in which the application would be developed.
  - Data persistence and consistency.
  - Server-side.
  - Client-side.

At first sight, it can be observed that these are mostly technical details that, as stated before, do require of a more extensive research since they are factors that will affect greatly on the performance of the web application and thus on the overall user experience.

### 3.2.1 Authentication possibilities

Authentication, or the process in which a user will enter some form of credentials which would confirm his or her identity from the server point of view in order to access some information restricted to other users, would become a critical step in the future versions of the application to ensure each professor would get their own exercises and that no other user could access, modify or delete these exercises.

At first, the proposition to add a simple user/password field seemed the most obvious and simple to complete: having the user input a user name (probably the email as it is common practice, for both practical and security reasons). The simplest way to have the user input these is by placing these two form items under a simple, single view that would only contain these elements, to keep the tool simple and unobtrusive, without any pop-ups or modals to get in the way, partially contributing to keep the interface as straight forward as possible.

A sample of the initial design for this view is shown in figure 3.2.

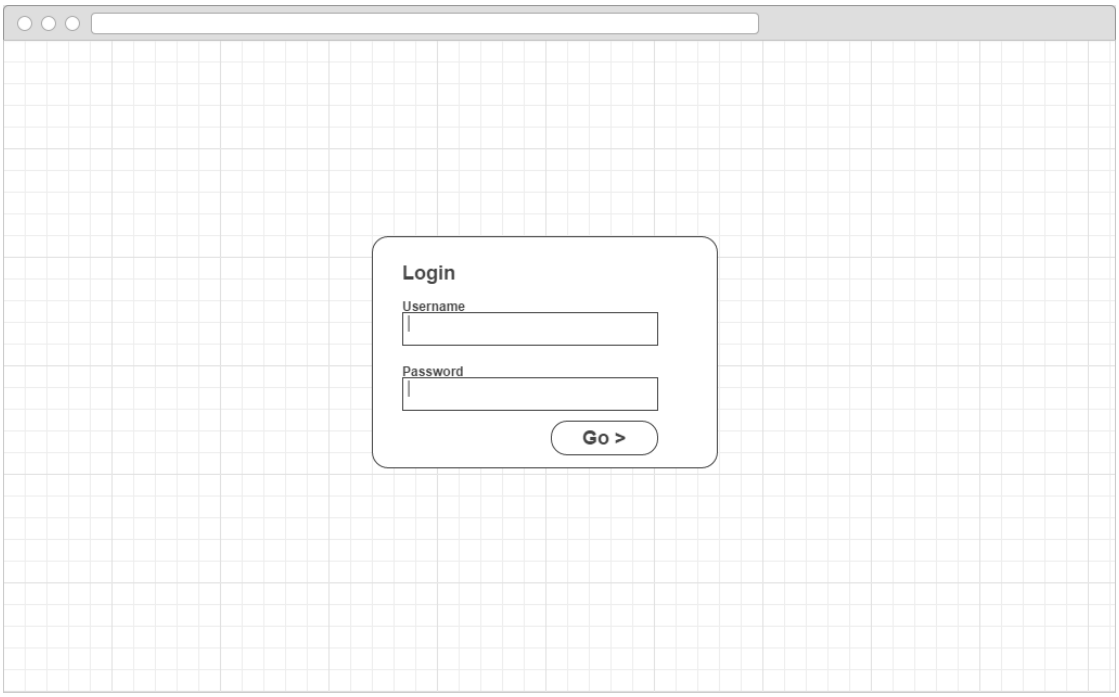


Figure 3.2: Login design

This view would also work for the second option that was presented to us: have the user log in and prove their credentials by logging in using their user name and password from an already existing LDAP service, namely the University's LDAP that all students, professors and staff members need to use in order to access the university's website internal tools, such as Aula Global (our custom implementation of Moodle) and Campus Global (Account management, where students can view and change all sorts of settings and information, such as timetables, payment information and more.

The idea of having a form of authentication, however, was left for a further, more advanced version of the application, as of now, having this form of log in would take time to implement and specially test and QA.

### 3.2.2 Management view

The management view was conceived as the view users would go as soon as any authentication was done, or directly to it in the event no authentication was added. This view was designed for users to see a summary of each exercise they have made, together with the option of editing the exercise and deleting it, all from the same place.

Figure 3.3 shows what the original design for this view was, a simple list of a few words from each exercise with two buttons per row, one for editing and one deleting the exercise.

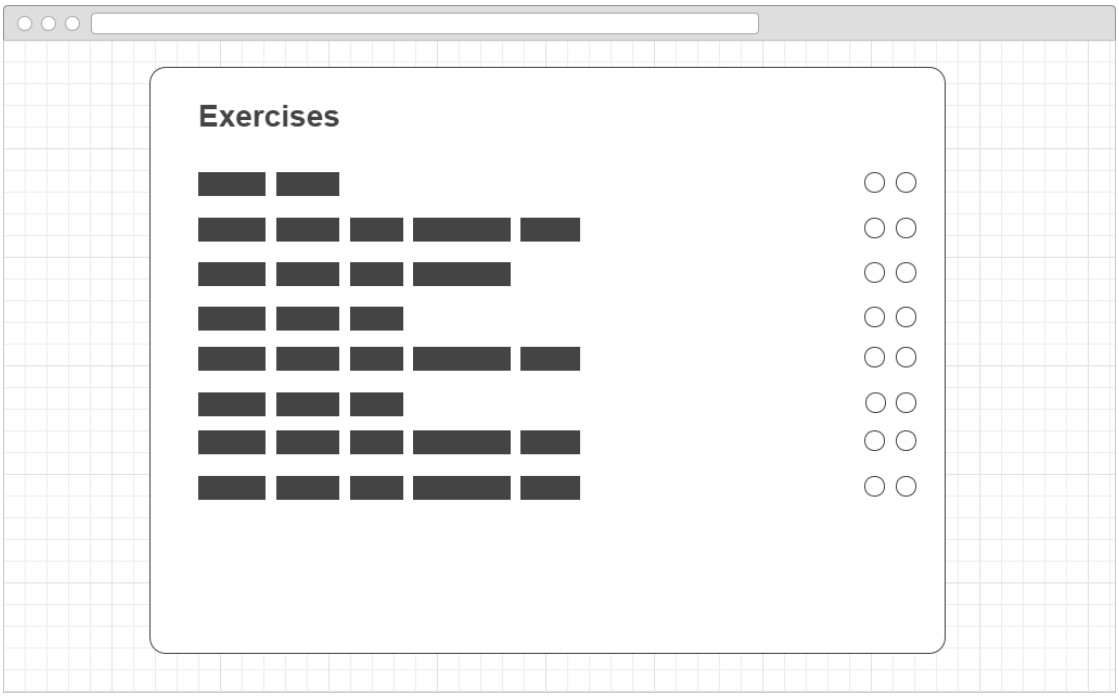


Figure 3.3: Initial management design

Much like authentication, this part of the project was decided to be left out for now, focusing on the main view, where the user could edit the exercise.

### 3.2.3 Data persistence and consistency

On the search for a definition of data persistence, I have found not better description than that of Wikipedia's, which states:

> In computer science, persistence refers to the characteristic of state that outlives the process that created it. Without this capability, state would only exist in RAM, and would be lost when this RAM loses power, such as a computer shutdown.
>
> This is achieved in practice by storing the state as data in non-volatile storage such as a hard drive or flash memory, most basically via serialization of the data to a storable format, and then saving the data to a file[13].

An application such as Genghis, where the user may take a long time in the creation of an exercise, must have some form of data persistence, be it in any of the forms above or more efficient forms of storage.

During the research on the different forms of data persistence, the following were the ones considered:

- **Files**: The simplest form of data persistence is to save the data into a binary file for later retrieval. The data is formatted in a specific format designed for easy retrieval. This method is, however, not suitable for our needs. Even if it technically can be used to store the exercise data, this method has a series of issues that helped us decide against its use, namely:
  - Not secure: Text files require further encryption that other forms of data storage already provide out of the box.
  - Slow: read and write operations are usually much slower than other forms of storage.
  - Concurrency: concurrency issues may arise, since the read and write of files is not managed by default in most systems and would require of a manual implementation of locks, a hard task specially when dealing with websites that may have several requests at once.
- **NoSQL**[14]: Non-relational databases have been used since the 60s, but its use did not become popular until early 2000s. These forms of databases have no relations between fields and store what is usually known as collections. A clear example of these collections is the popular NoSQL MongoDB or Cassandra. These characteristics make it ideal for Genghis' authoring tool but it was decided not to use them for two main reasons:
  - We would almost certainly need to use a SQL database for user authentication, and having two separate sets of databases was deemed too farfetched for the extent of this project. This could, however, be added as a further development to the tool, since it would most likely add speed to the application.
  - Laravel, the server side PHP framework that was used, offers an Object-relational mapping system which simplifies the work relating with databases tremendously. This together with migrations made clear the decision to stick with a single relational database.
- **Relational databases**: Relational databases are characterized (unlike its NoSQL counterpart) for having relationships between their fields. For instance a table that held the exercise information could get the authors data from another table holding the user's information. This fact, together with all the advantages that using Laravel and an ORM bring us were the key elements in choosing this form of database.

Once the decision to use SQL was taken, the next was to decide on a specific form of SQL from all the available forms of relational databases out there. We narrowed our choices to the following:

- **SQLite**: A form of SQL where all the data is stored inside a *.sqlite file.
- **MySQL**: One of the most popular open source SQL databases.
- **PostgreSQL**: A very powerful SQL database engine, but far less popular than MySQL
- **Microsoft SQL server**: Microsoft's implementation of relational SQL databases.

From these options MySQL was chosen, given its popularity, the very well done documentation and previous experience with its syntax.

### 3.2.4 Server-side

Note: the decision of which database to use was encouraged partially by the choice made in this section, both of them came together at once since they are very closely related, as it will be demonstrated shortly with the Model View Controller architectural pattern.

Server-side is what is known as the programming in charge of handling the client's request and returning the proper response, according to a predefined logic. Thorough the years there have been various forms of tackling the architecture behind a website's back end but the most commonly used nowadays is the Model-View-Controller pattern.

Taken from Microsoft's knowledge base, the Model-View-Controller is defined as:

> The Model-View-Controller (MVC) pattern separates the modeling of the domain, the presentation, and the actions based on user input into three separate classes:
>
> - **Model**: The model manages the behavior and data of the application domain, responds to requests for information about its state (usually from the view), and responds to instructions to change state (usually from the controller).
> - **View**: The view manages the display of information.
> - **Controller**: The controller interprets the mouse and keyboard inputs from the user, informing the model and/or the view to change as appropriate.

This definition is explained further below, with the help of figure 3.4.

As it was detailed before, the Model-View-Controller is an architectural pattern created and designed with the idea of making the logic behind websites something separated from the presentation and the business logic. This is done by means of the three parts that compose the Model-View-Controller:

- **Model**: The model takes care of the business logic, it is the part of the Model-View-Controller in charge of handling the data storage and persistence, as well as any mapping

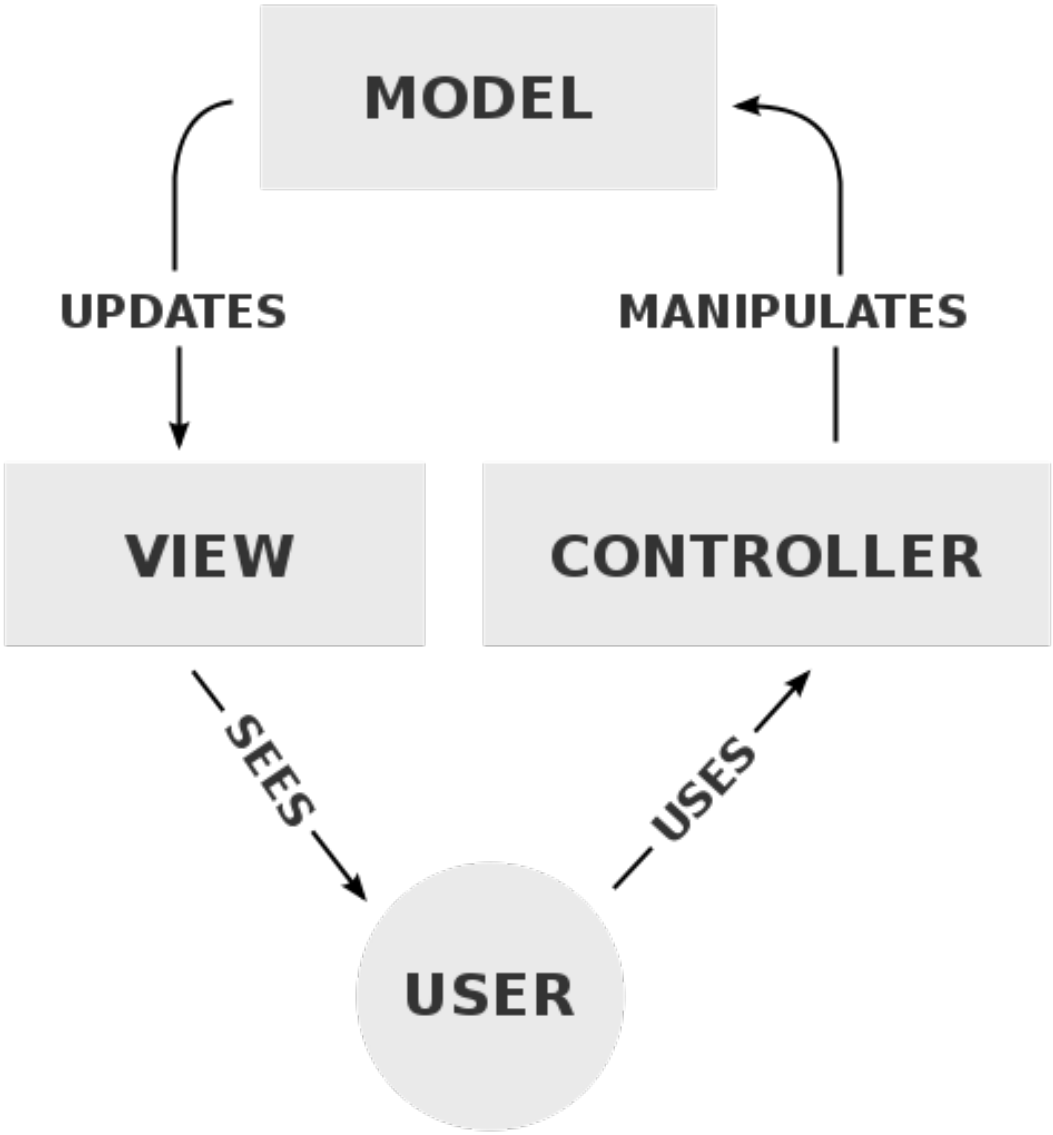


Figure 3.4: The model - view - controller schematic[15]

done to databases or APIs (including API wrappers). In our case Genghis' authoring tool models would be composed of connections to the MySQL database and the mapping of all the objects to custom models, such as an "exercise" model or a "user" model via Object-relational mapping or manual creation.

- **Controller**: The controller is in charge of the logic in the website. It is the part that takes user input and, by means of a set of predefined rules, returns (most of the time) a response to the user. The controllers must never connect to databases directly and should use instantiated objects of the models created at the Model part of the Model-View-Controller.
- **View**: The views take care of the presentation of data to the end user. This is usually done by generating the HTML code together with the processed data the controller has sent to the view, usually by means of engines like Razor for Microsoft's MVC and Blades in Laravel. As little logic as possible should be placed in the view, as it will greatly hinder the speed of the site. In Genghis, the Views could be "/list" for listing the exercises (management view) and "/edit" for editing an exercise.

It is crucial to understand that the three elements composing the Model-View-Controller design pattern have very specific tasks and any form of changing or mixing roles should be avoided at all costs, since following the pattern usually leads to an increase in not only code organization, but efficiency as well. The most common mistakes when dealing with Model-View-Controller are usually:

1. Add business logic to the controllers, specially database queries.
2. Have the views do more processing than they are required to. As much processing and filtering as possible must be done in the Controllers, so views receive only a small set of only the required data.

The decision to use MVC was followed by deciding which form of programming we would use for the authoring tool. This decision was split between implementing our own MVC or using an already existing framework.

There were several options available when choosing framework but based on previous knowledge on the programming language the framework was based on and the size of the community (as most of this projects are open source, the bigger the community the simpler it is to find answers should any problem arise), we narrowed down our choices to the following two:

#### Ruby on Rails

Ruby on Rails[16] (usually known as Rails) is a web framework based on the popular scripting language Ruby. This framework's initial release was back on 2005, when its creator, David Heinemeier Hansson decided to openly share the open source project. Its popularity exploded back in 2007 when Apple decided to include Ruby on Rails with their new operating system OSX 10.5 Leopard.

Below are shown what the advantages and disadvantages of using Ruby on Rails for our project were:

- Advantages

  1. **Modern**: It is a fairly recent and well updated framework designed with many of he problems other programming languages had in mind.
  2. **Flexible**: Supports more programming paradigms, not only OOP.
  3. **Simple syntax**: The syntax Rails borrowed from Ruby is very simple to understand, although it may take some time to learn.

- Disadvantages

  1. **Community**: Although growing in numbers, Ruby on Rails has a smaller community than other programming languages and frameworks.
  2. **Deployment limitations**: Finding a place to host a Ruby on Rails project is not an easy task, at least compared to frameworks in other languages such as PHP.
  3. **Slower**: Although there has been major improvements in efficiency in the last few versions, Ruby on Rails still lags behind in efficiency.
  4. **Prior knowledge of the language**: As the only programmer, I was not completely comfortable with programming in Ruby the entire project.

In our case, the disadvantages outweighed the advantages, since the project would be hosted in a University's machine that was shared with other users and taking the performance hit a Ruby on Rails application would do at the time was not an option. Also, having to learn not only the framework, but specialize in the language as well would greatly hinder the progress of the project, which already had a deadline.

#### Laravel

Laravel[17] is a PHP based framework "for web artisans", it was originally built by Taylor Otwell and released in early 2012. As of today it is one of the most popular frameworks for web development and the most popular PHP framework, surpassing Synfony.

Below is an extract from Laravel's philosophy which illustrates its purpose and further describes what its creator had in mind when creating it.

> [...] Laravel aims to make the development process a pleasing one for the developer without sacrificing application functionality. Happy developers make the best code. To this end, we've attempted to combine the very best of what we have seen in other web frameworks, including frameworks implemented in other languages, such as Ruby on Rails, ASP.NET MVC, and Sinatra. [...]

- Advantages

  1. **Modern**: Much like rails, it is a very recent framework and as it was explained in Laravel's philosophy above, it was built "combining the very best of what we have seen in other web frameworks".

2. **Deployment**: Unlike rails, most online hosts support PHP and Laravel, and a simple Apache server with minor tweaks can also run it with little overhead.
3. **Simple syntax**: PHP is also a very simple programming language.
4. **Prior knowledge of the language**: I have had worked on other PHP projects before, and although I had never used a PHP framework before, the process was probably going to be simpler when starting from an already known language.
5. **Eloquent ORM**: The magnificent Object-relational mapping Laravel includes is incredibly fast and that together with migrations made Model handling very easy.

- Disadvantages

1. **Community**: Growing very rapidly, but since Laravel was released two years ago, other frameworks have had more time to create more communities.
2. **Newer framework**: The very early versions of Laravel still had some minor issues and bugs that were quickly fixed by the community.

Given these choices: and the similarities both projects offered such as great, active communities, open source projects, easy languages and lots of tools for model mapping and view engines, the decision taken was to use Laravel, mostly because of the simplicity it offered and the swiftness not having to learn a new language gave us.

This, however was not the final choice since time pressed and there was a deadline to meet, the final decision was to implement our own simple MVC in PHP, making a small test website for professors to use and give feedback on, while learning how Laravel worked and implement the website using the framework in a future version.

### 3.2.5 Client-side

The Model-View-Controller is, in itself, enough for having a website function correctly on the server side: all the processing is done in the server and thus far none in the client.

There are times, however, when user actions in the view require processing being done in the browser, for example animations, form validation before submission so as to avoid unnecessary requests to the server... etc.

Genghis' authoring tool would have to have this client side processing for some various tasks, which before the project started were identified as:

1. Exercise deletion via Asynchronous requesting.
2. Variable handling, specially hiding unnecessary fields when variable types change.
3. The WYSIWYG editor.

For this purpose, the following three JavaScript libraries were chosen: jQuery, jQuery UI and Twitter's Bootstrap JavaScript libraries.

jQuery[18] is one of the most common and widely used JavaScript libraries in the world. Originally released in mid-2006, jQuery is a DOM traversal library built with the intent of making JavaScript simpler and clearer for its users by simplifying and shortening the notation to do vanilla JavaScript and adding other methods to complement it.

A comparison of some of the most common tasks performed in vanilla JavaScript compared to their jQuery counterpart is shown below:

1. Selecting a div by ID:

```
document.getElementById('myId');
```

vs

```
$('#myId');
```

2. Change the body's font color:

```
    var changeFontSize = Function (color) {
        Document.body.style.font-color = color;
    }

    Onload=changeFontSize(blue);
```

vs

```
$('body').css('font-color', 'blue');
```

3. Ajax request:

```
    var req = new XMLHttpRequest();
    req.open("POST", "genghis", true);
    req.onreadystatechange = function () {
        if (req.readyState != 4 || req.status != 200) return;
        alert(req.responseText);
    };
req.send("param1=1&param2=2");
```

vs

```
    $.ajax({
        url: "genghis",
        type: "POST",
        data: {param1: 1, param2: 2},
        success: function(receivedData) {
            $('#resultDiv').text(receivedData);
        }
    });
```

The simplicity it add to the project's front end processing also comes at a cost, since jQuery is outperformed by vanilla JavaScript in performance tests, although the benefits it offers outweigh the performance difference.

jQuery UI[19] and Twitter Bootstrap[20] are both JavaScript libraries that complemented their CSS counterparts and their main purpose was to help in the development of the Views, adding extra value to the CSS like tabs (shown in figure 3.5) that could be used for the preview pane, modals (shown in figure 3.6) for extra options or variable edition or even draggable elements (figure 3.7) for the management view.

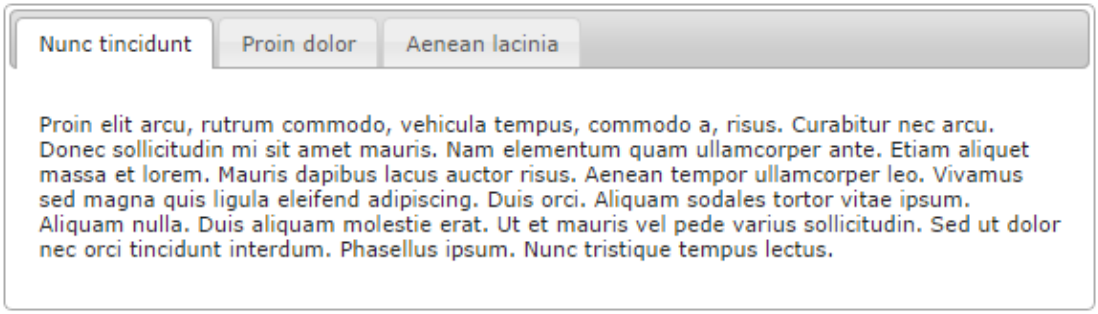


Figure 3.5: Sample jQueryUI tabs

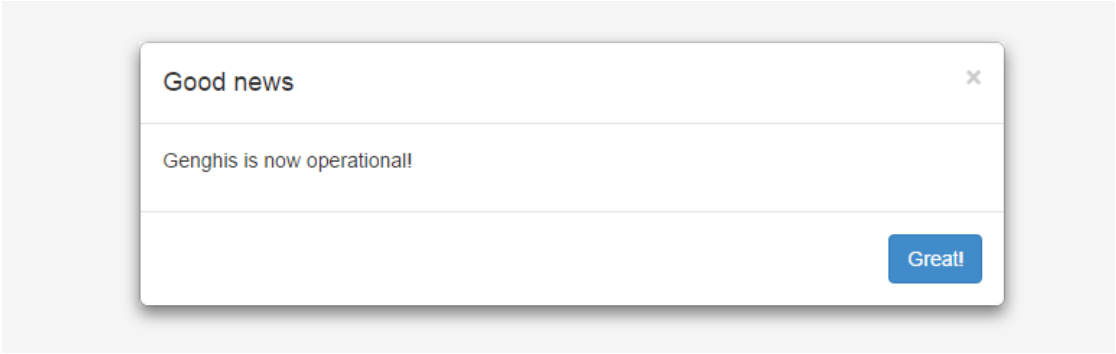


Figure 3.6: Sample Bootstrap modal

Since this was to be the early version of the Genghis authoring tool, the default CSS were left to use for these tools (jQueryUI and Twitter Bootstrap) until the first version was released and feedback was obtained from professors.

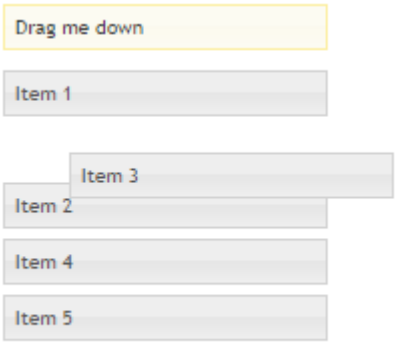


Figure 3.7: jQueryUI Draggable elements

# Chapter 4

# Development

Once the first stage of design was finished and a clear plan of what the application would look like, both visually and in terms of code, it was time to step in the second stage of the process: development.

The development of Genghis' authoring tool could be split into several steps and although some took place together (since the development of a web application using MVC has to sometimes have advance in Controllers and Views in parallel (and also front-end), the steps described below summarize the development of the first beta version of Genghis' tool in chronological order.

1. **The initial setup**: Where the machines to be used were prepared and the corresponding services prepared.
2. **Database preparation and modeling**: Deciding where to set the database up and what tables to use.
3. **Preparing the Views**: the preparation of the master layout, as well as the structure of what the page would look like once finished.
4. **Creating the controllers**: Separating the tasks into controllers.
5. **Front end development**: All the JavaScript used.
6. **Rework of the view**: after almost the entire development was completed for the first beta, some changes were made to the Views.

## 4.1 Early stages

Before starting to work, the decision of where to host Genghis was still to be made.

It was decided that we would follow a system architecture similar to what is depicted in figure 4.1:

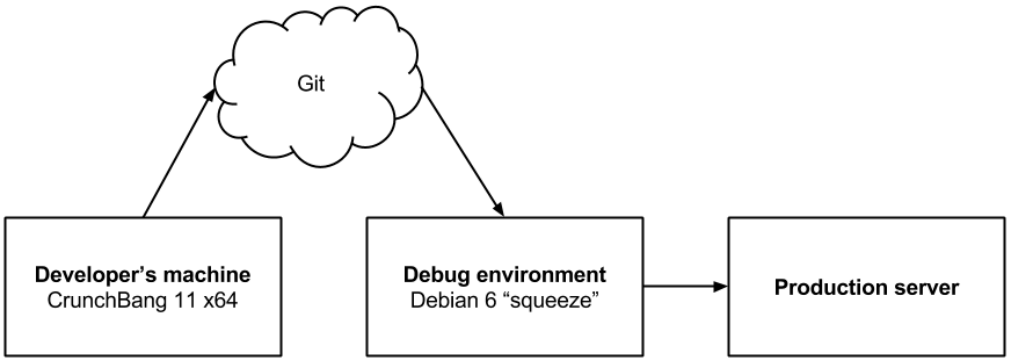


Figure 4.1: Machine schematic

The main idea was to have the developers machine work on the code and Genghis' features, upload the changes to a version control system, such as git or subversion. Main changes to a master branch would be downloaded to the debug server. Finally once the release date approached, the most stable code would be taken to the production server, unknown at the time.

Git[21] was chosen as a version control system for Genghis for the following reasons:

1. Popular.
2. Great community.
3. Open Source GUI tools to help with branch visualization, merges and rebases.
4. ZSH's git plugin, for shortening commands and include new, more powerful aliases.

So we set up a local git project in the developer's machine, as well as its remote counterpart online, so there would always be a version of the code online, minimizing the possibilities of losing the code. An empty initial commit was made to ensure everything was properly set up.

After asking the University's IT department, they granted us access to a dual core machine running Debian 6, a very common and stable Linux distribution. This machine will serve as the main debug version of the Genghis tool, where interested parties could have a look at the latest most stable version of the code that would only be uploaded when a basic quality assurance was done.

Even though we were granted a user and a password, SSH tunneling was changed to work using public keys, a more secure approach to this protocol. Mainly because the username was given and could not be changed, increasing the risk of someone finding out the user password and logging in to the debug machine, with potentially catastrophic results.

The debug machine was, as mentioned before, a Debian 6 machine. To run Genghis using

PHP, we asked the IT department (since we were not granted *sudo* privileges) to install the latest version of Apache stable, python, vim, tmux, zsh and git for the following reasons:

- Apache: To run the necessary PHP files for the server side of the tool. This would also be used to host any resources such as pictures, images, JavaScript files and CSS files.
- Python: Used to run the development version of the previewer. The previewer were a set of files that were given in the Khan Academy exercise module whose purpose was to convert Khan Academy's HTML exercise files into actual HTML that could be rendered by a browser.
- vim: For quick editing of files in the server, like files that held connection strings that could not be uploaded to git for security reasons.
- tmux: for holding sessions. Initially there was no possibility to have Cron jobs so it was decided to run tmux as an alternative to screen while this problem was fixed.
- Git: For downloading the latest, most stable version of the master branch.

## 4.2 Databases

As mentioned in the Post-Brainstorm investigation section, a decision was made to use MySQL databases in Genghis, to store the exercises' data and variables.

There were two databases initially set up for this project, one was to become the debug database, where tests would be done and data could be added manually to the database via queries, one the other hand there would be a production database, a more secure replica of the debug database, that should only be filled with real data from real users, and never by hand. This is represented in figure 4.2, which is an enhanced version of the schematic in figure 4.1.

As to how the tables are laid in these Databases, an analysis of the main "objects" of Genghis' exercises was made. By objects it is meant the entities that make up Genghis or, basically, what would later become Models once the implementation in Laravel took place. And a conclusion was made that, for the first initial version, there were mainly 3 objects:

1. **Variable**: A variable would be an object assigned to an exercise and would hold various information relating to it, such as name and range.
2. **Hint**: Like a variable, it would be assigned to an exercise and would be mostly composed of the hint's text.
3. **Exercise**: The exercise itself, would hold all its information, including statement and author.

The next logical step was to map these objects into 3 tables, one per object. The problem was that one of the professors' requirements were to have both integers and floats for exercise variables, rising the problem that extra field was necessary for floats: the step, meaning the step in which the random value the variable could take would jump from i.e. if a variable was set

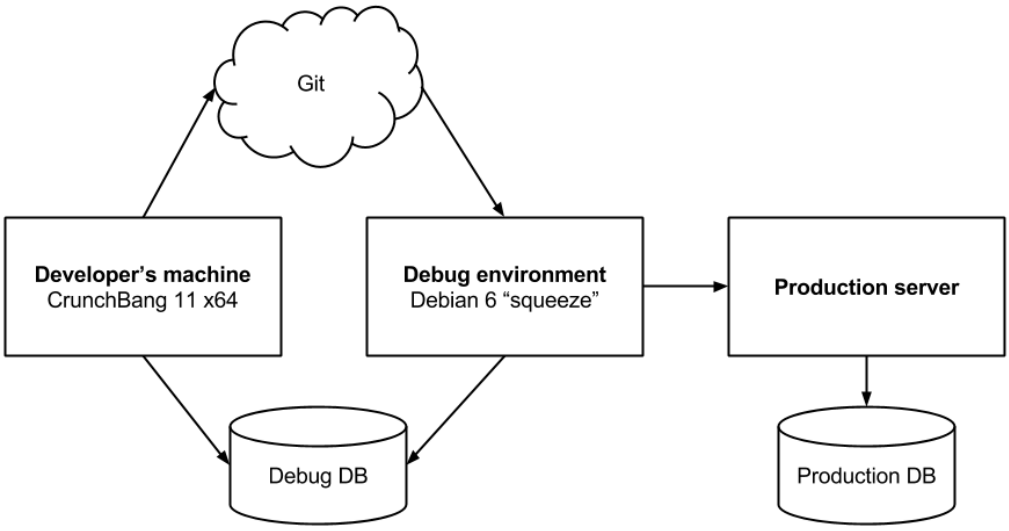


Figure 4.2: Machine Schematics with Databases

from 2 to 6 in steps of 0.5, the random values possible would be 2, 2.5, 3, 3.5, 4, 4.5, 5, 5.5. and 6.

This minor inconvenience led to the separation of the possible variable table into 3, a table for the range and step of floating point variables, a similar table without the step for integers and finally, a variable table for the common fields, like the exercise they belonged.

In the end, the database was created with five tables, namely:

1. khan_question
    - **question_id**: The table's Primary Key.
    - **question_author**: A string that would store the author's username. This field was made with a possible future change to refer to whatever authentication method was selected.
    - **question_course**: An integer in charge of storing the course ID of the exercise. This was a requirement by the platform it would be stored in the future.
    - **question_title**: A string with the title of the exercise.
    - **question_statement**: A string of up to a thousand characters.
    - **question_solution**
    - **question_round**: A Double meant to hold the rounding of the solution, as the operations could result an irrational number for a solution, which would make it hard for students to solve the problem.
    - **question_error**: The error margin a student can have and still have the question set as correct.

- **question_check**: A check of whether the user wants the solution accepted displayed in the exercise body, only for testing purposes.

2. khan_hint
   - **hint_id**: The table's Primary Key.
   - **hint_question**: The primary key of the question it belongs to.
   - **hint_text**: A 1000 word string holding the hint itself.
   - **hint_order**: Hints can be ordered, so here the order in which they should be displayed is stored.
3. khan_variable
   - **variable_id**: The table's Primary Key.
   - **variable_question**: The primary key of the question it belongs to.
   - **variable_name**: A 255 character string storing the variable's name.
   - **variable_type**: An Enum denoting whether the variable is a floating point variable or, on the other hand, an integer.
4. khan_variable_integer
   - **integer_id**: The table's Primary Key.
   - **integer_variable**: The primary key of the variable these parameters belong to.
   - **integer_min**: An integer with the minimum value the variable can get.
   - **integer_max**: An integer storing the maximum value the variable can get.
5. khan_variable_float
   - **float_id**: The table's Primary Key.
   - **float_variable**: The primary key of the variable these parameters belong to.
   - **float_min**: An floating point number with the minimum value the variable can get.
   - **float_max**: An floating point number storing the upper limit the random number generation can get.
   - **float_step**: An floating point number storing the step at which the random number can be generated (0.5 in the case above)

To better represent these objects, a schema of the Database is presented in figure 4.3, including the relationships among its tables.

## 4.3 View scaffolding

Once the database was set up it was time to prepare Genghis' authoring tool Views. In the original plan, only one View was to be made, the one designed to edit the exercise.

To do so, the following libraries and files were added to the project's files:

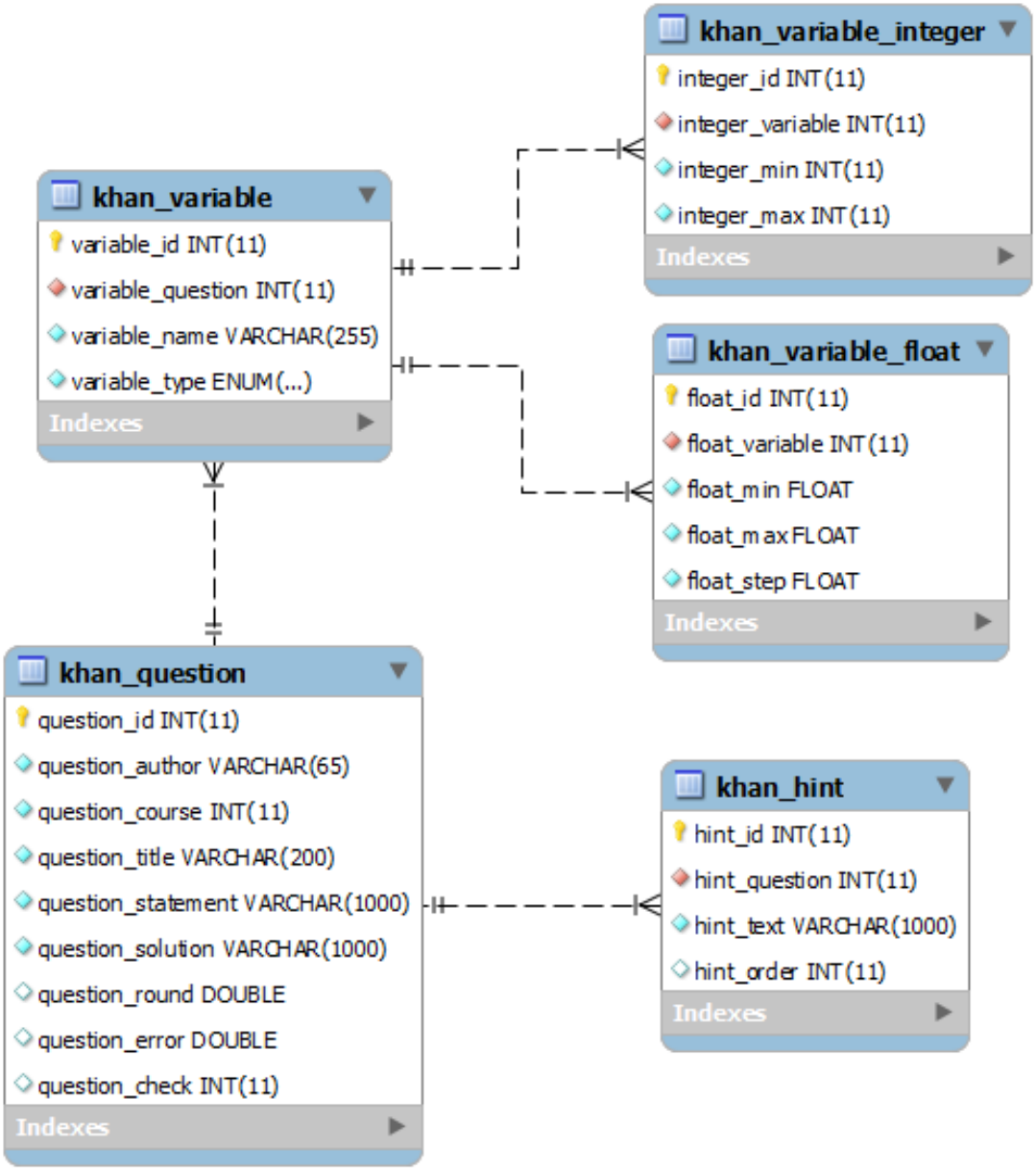


Figure 4.3: Database Schematics

- The twitter bootstrap
- jQuery
- jQueryUI
- A custom.css file

The custom.css file was created to hold custom CSS rules and not alter other libraries. Having the HTML import the CSS libraries before custom.css ensured that most rules would be applied correctly.

The main view was structured as follows:

1. Meta tags
2. Title
3. CSS imports
4. Main body
5. JavaScript imports
6. custom JavaScript

The order in which this view is structured is also important for various reasons: first, CSS requests are asynchronous and non-blocking, making them the ideal files to be requested and downloaded first. JavaScript on the other hand is blocking and until a file is completely downloaded and parsed, the next download will not occur. This is why the JavaScript requests are performed after the main body, to ensure a fast load and minimum content is ready to be presented to the user while the rest of the files download[22].

Secondly, the custom JavaScript that was not on files (mostly initialization of resources, has to be put after those files have been downloaded or the browser will raise an exception, as the object will not be found.

The first design for the view was based on a sample of the Twitter Bootstrap, in which, using their grid system with 12 columns, the simple structure of the exercise editor could be prepared.

This grid system is one of the Twitter Bootstrap's most important and used features. The developers designed a series of classes to be added to ¡div¿ tags in order to restrict and control the width of these, even in responsive mode.

To do so, they divided the width of the body into 12 separate columns (a deliberate choice probably, as they are well divided into blocks of 1, 2, 3, 4, 6 and 12) and by using a class "row", structuring a website is a simple matter of assigning the correct number of columns per div, as shown in figure 4.4

By using this method, obtaining the first version of the main view of the tool as a simple matter of reading the documentation and following the design mockup in figure 3.1 with slight variations, such as changing the width of the exercise preview pane to leave more room to the editing.

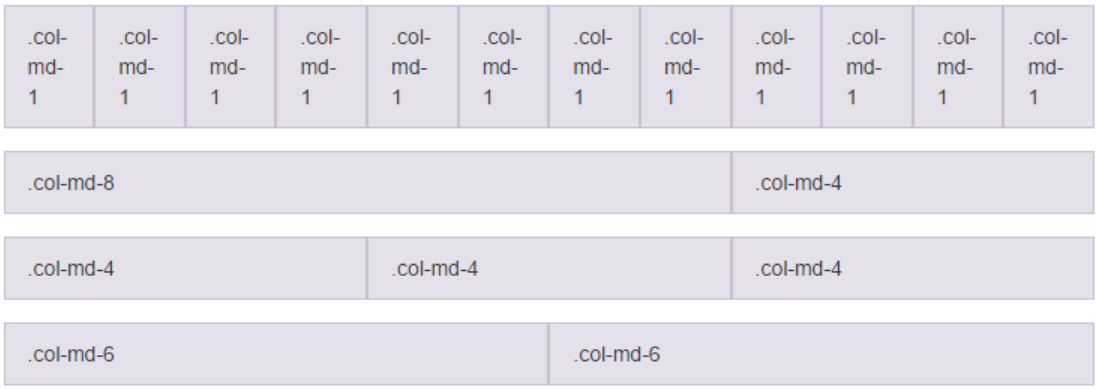


Figure 4.4: An example of how Bootstrap's column system works

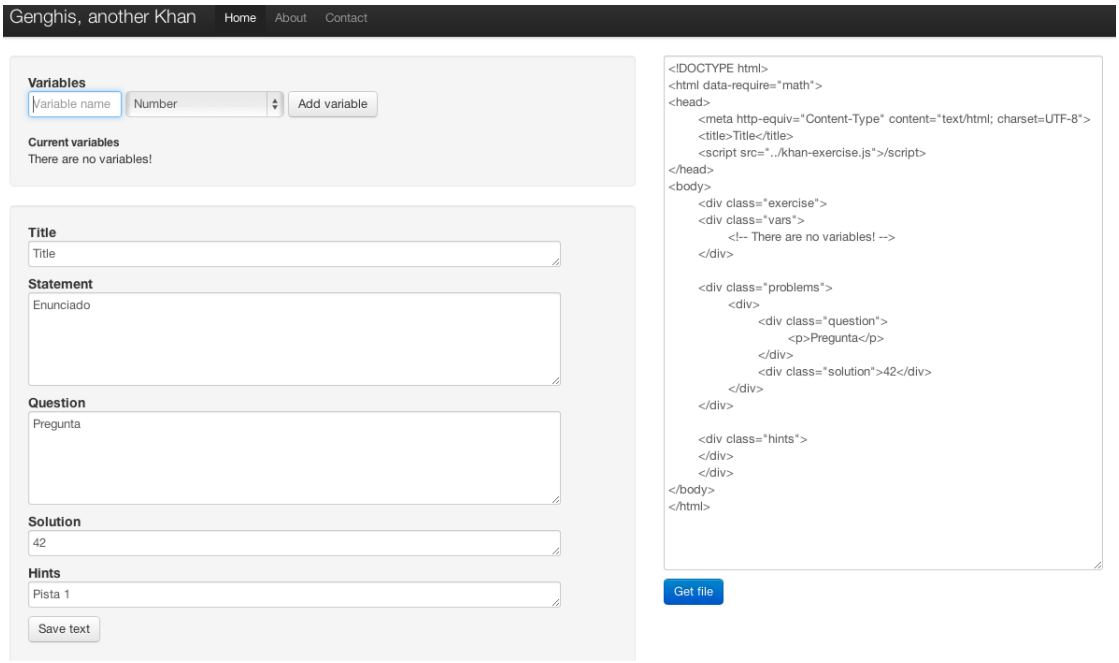


Figure 4.5: First design of the edit exercise View

## 4.4 Controllers

The controllers were one of the most challenging parts of the development of the Genghis tool. At first and for the initial version only, these were split as follows:

1. Variable edition and handling
2. Hint edition and handling.
3. Parser
4. Exercise controller.

These sets of controllers were the core logic behind Genghis and its authoring tool, each of them had a specific set of tasks, mostly related with input validation and Model storage, which are detailed below:

### 4.4.1 Variable Controller

This controller was in charge of creating, deleting and editing variables, first validating the input just in case something was go wrong or any input was tampered with during request.

The variable controller's methods followed a specific set of instructions, detailed as follows in the following UML diagram:

### 4.4.2 Hint controller

The hint controller was designed to be very similar to the variable controller, its primary functions were:

1. Add hints
2. Delete hints
3. Edit hints

And the methodology used was almost identical to the one used in the variable controller, this is the reason the UML diagrams have not been included in this paper.

### 4.4.3 Parser

The parser was the controller in charge of taking the exercise's parameters, variables and hints and turning them into Khan Academy styled HTML.

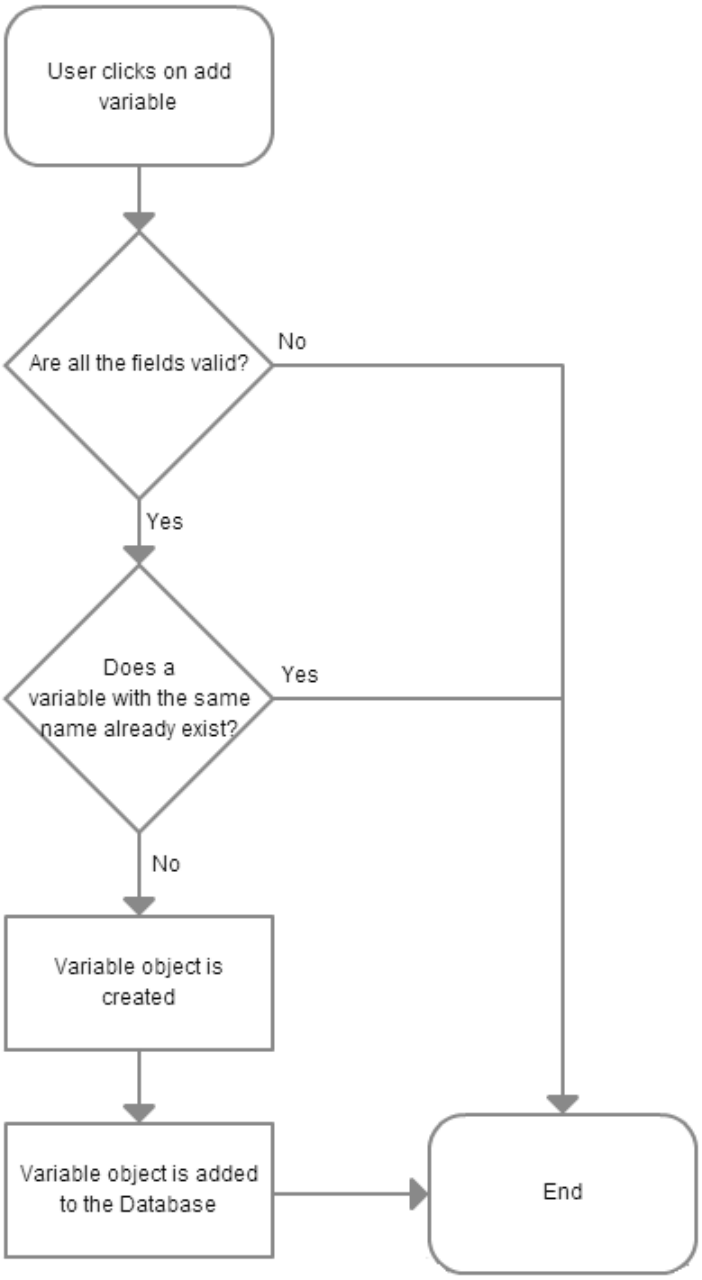


Figure 4.6: Adding a variable

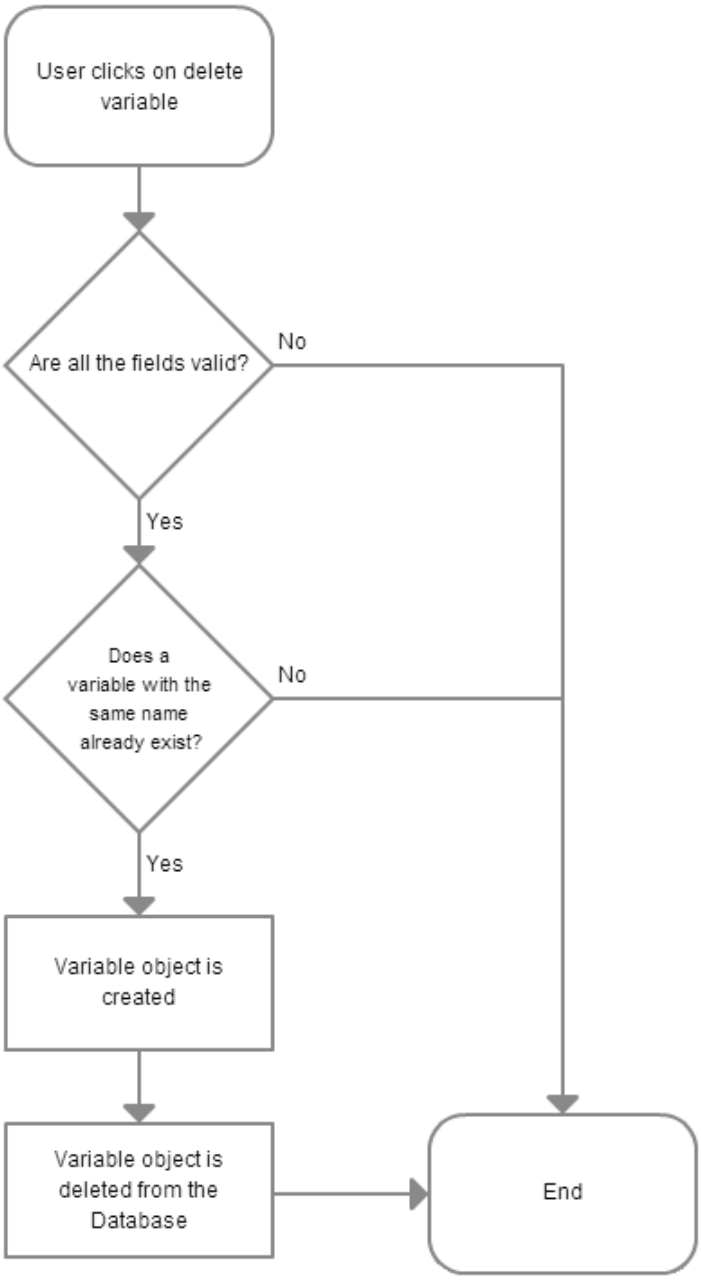


Figure 4.7: Deleting a variable

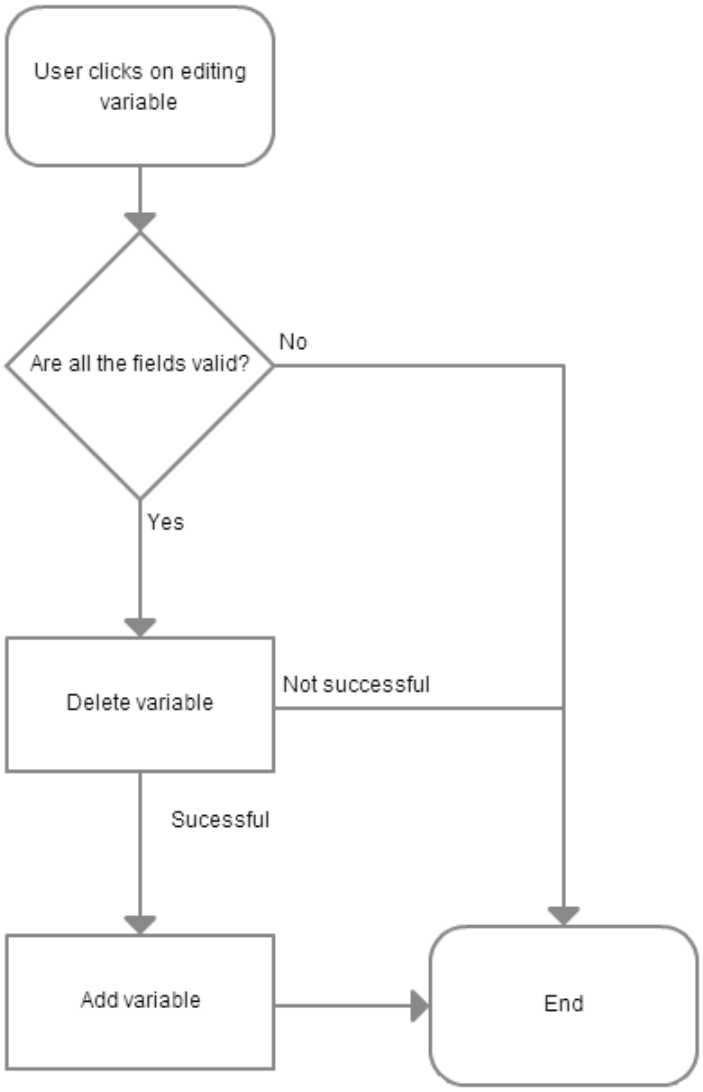


Figure 4.8: Editing a variable

The task was initially set to be done by a manual parser, taking the content from the models and manually creating the string that was later saved to a file for the exercise engine to render into a complete exercise for final use. This practice however changed promptly as the task got more tedious and the methods more resource consuming. The generation of an XML string manually by means of concatenation and string formatting is a costly one, at least compared to using a XML builder like SimpleXML, a PHP extension that can be used to generate XML dynamically. An example of this would be:

```
$exercise = new SimpleXMLElement('<html/>');

for ($i = 1; $i <= $variableCount; ++$i) {
    $variable = $exercise->addChild('variable', $i);
    $variable->addChild('minimum', $model->getVariable($i));
}

print($exercise->asXML());
```

This practice is not only cleaner, but also more efficient and closer to a more Object Oriented Programming approach.

The purpose of this controller, as it was stated before is to generate Khan Academy styled exercises, like the one shown in page 53, a sample exercise from the parsing engine:

```
<!DOCTYPE html>
<html data-require="math graphie graphie-helpers-arithmetic word-problems">
<head>
    <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
    <title>Add within 5</title>
    <script data-main="../local-only/main" src="../local-only/require.js"></script>
</head>
<body>
    <div class="exercise">
        <div class="vars">
            <var id="A">randRange(1, 4)</var>
            <var id="B">randRange(1, 5 - A)</var>
        </div>

        <div class="problems">
            <div>
                <div class="question">
                    <div class="graphie">
                        init({
                            range: [[0, 12], [-1, 1]]
                        });

                        var equation = rand(2) === 0 ?
                            "\\Huge{\\blue{" + A + "} + \\green{" + B + "} = {?}}" :
                            "\\Huge{{?} = \\blue{" + A + "} + \\green{" + B + "}}"

                        label([0, 0], equation, "right");
                    </div>
                </div>
                <div class="solution" data-forms="integer"><var>A + B</var></div>
                <div class="hints">
                    <div class="graphie" style="float: left;">
                        drawCircles(A, BLUE);
                    </div>
                    <div class="graphie" style="float: left;">
                        drawCircles(B, GREEN);
                    </div>
```

```
                    <p style="clear: left;">
                        <span data-if="isSingular(A)">
                            There is <code>\blue{<var>A</var>}</code> blue dot.
                        </span>
                        <span data-else="">
                            There are <code>\blue{<var>A</var>}</code> blue dots.
                        </span>
                        <span data-if="isSingular(B)">
                            There is <code>\green{<var>B</var>}</code> green dot.
                        </span>
                        <span data-else="">
                            There are <code>\green{<var>B</var>}</code> green dots.
                        </span>
                    </p>
                    <p>There are a total of <var>A + B</var> dots.</p>
                </div>
            </div>
        </div>
    </div>
</body>
</html>
```

## 4.5 Front-end

The first version of the Genghis authoring tool was developed to work with little JavaScript. As explained before, only the initialization of external modules such as tabs or modals were included in the code, together with two exceptions:

### 4.5.1 JavaScript for variable deletion

Although initially set to be done one GET requests, the deletion of variables was changed to work from AJAX requests as well. Since the method in the controller was not create to distinguish between this, the only change to be done was to:

1. Remove the form element from the HTML.
2. Prepare an AJAX request.
3. Have the request be triggered on click of the deletion button.
4. Remove the variable row from the view every time the AJAX request is successful.

The form was removed as leaving the button to trigger a form will cause a redirect on the website, effectively rendering the AJAX process a failure. There was an option on JavaScript to trigger the event on the form's submission and simply prevent default behavior, triggering the AJAX request then. This was, however, much more complicated than the actual solution taken, which was to call the AJAX request on click, done using jQuery with the following line:

```
$('#variableContainer').on(
    'click',
    '.variableRow',
    deleteVariable($this.attr('id')));
```

Delegates had to be used since the load of this event is triggered on the *onload* of the DOM, making it impossible to know which variables are there initially, this way the event will trigger to any element with the class *variableRow* under the element with ID *variableContainer*.

Delegates are much less specific than direct event triggers but the loss in efficiency is minimal and the gain in functionality once again, outweighs any issues it may rise.

### 4.5.2 TinyMCE

There was one crucial part left in the creation of the Genghis exercise edition view: The editor.

Since the early stages, it was part of the planning to add a What You See Is What You Get Editor (WYSIWYG) to help professors create more visually appealing exercises by including basic tools such as text bolding, italics, enumerations, tables and finally, an equation editor.

For this task, an extensive research was carried to find the most complete, simple of use and well documented WYSIWYG editor. After narrowing down our choices, the decision taken was to use TinyMCE[23], a very powerful and modular JavaScript library that offered a great deal of plugins and customizable extensions. This editor was chosen also because is the same editor professors use every day in Moodle, and having familiar items on the web page may help simplifying the use of the tool, thus improving user experience.

The editor was configured to have the following options, as depicted in figure 4.9:

- Save and clear the field
- Text options: Bold, italics, underline and strike-though
- Alignment options
- Font configuration, size, family and text type.
- Copy and several format pasting possibilities, depending on the source.
- Search and replace.
- Enumeration, both ordered and unordered.
- Tab options.
- Image insertion.
- Date and time insertion.
- Text color and highlight.
- Table creation and editing.
- Greek symbols and icons.
- Movie embedding.
- Formula insertion, shown in figure4.10.

- Graphing possibilities.
- Variable handling, shown in figure4.11.

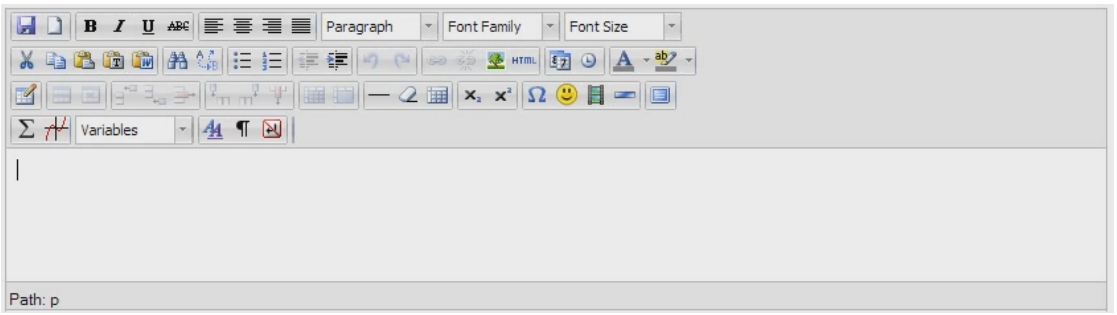


Figure 4.9: TinyMCE custom configuration

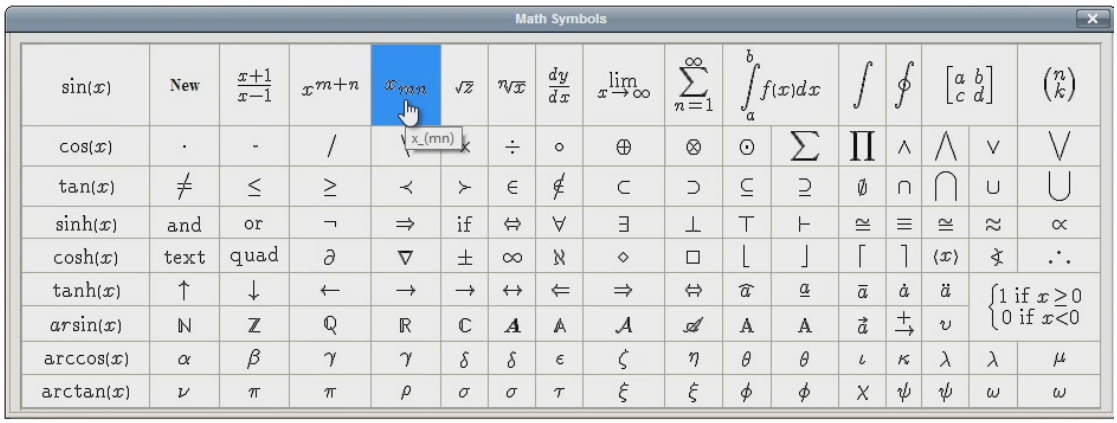


Figure 4.10: TinyMCE LaTeX formula insertion table

## 4.6 Rework of the view

Before the release of the first version of the tool, the team in charge of the platform where Genghis would be placed on, kindly sent us a set of CSS files and rules to use, in order to try and homogenize the look of the entire platform.

The CSS classes were very easy to implement and in a few hours, all the colors and gradients of the tool ha change. The only issue was that, in order to use the same body width as the rest of the platform, the width had to be limited for Genghis' authoring tool as well. This triggered the decision to use tabs for the exercise preview and led to the use of collapsible divs to store the different sections and editors of the exercise. The final result is shown in figure 4.12 and figure 4.13.

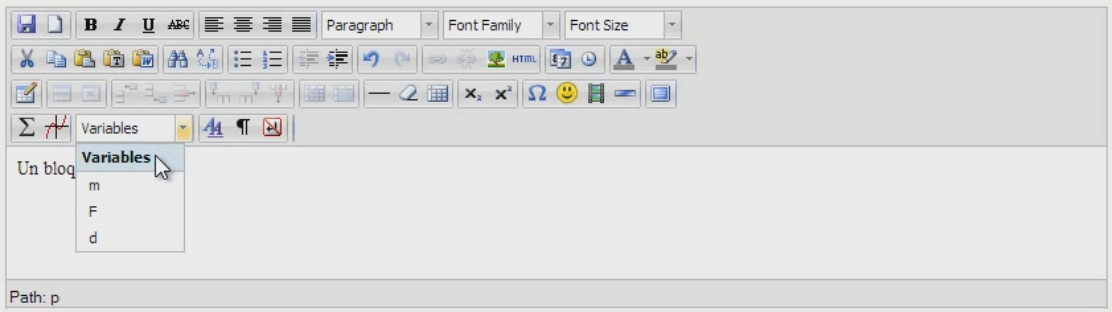


Figure 4.11: TinyMCE custom variable dropdown

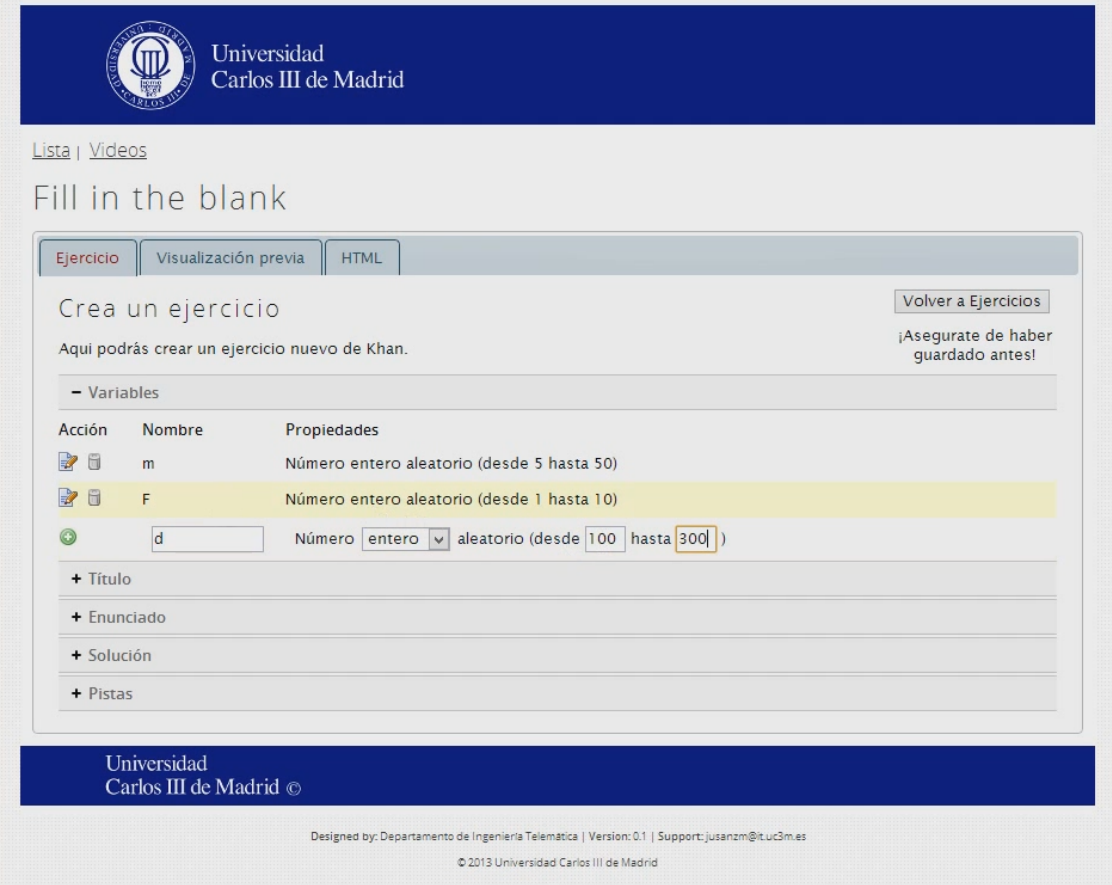


Figure 4.12: Final design

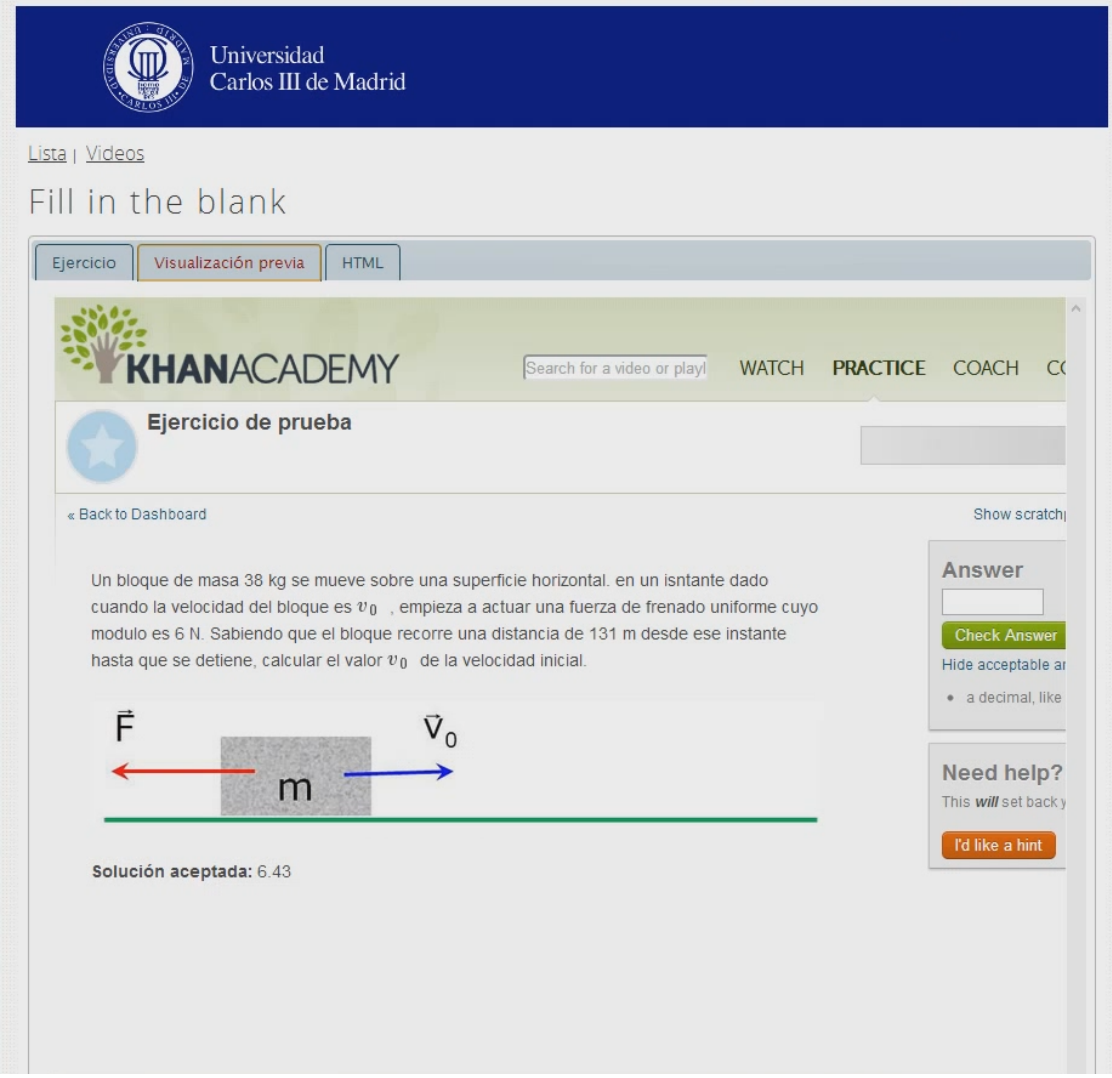


Figure 4.13: Preview tab of the final design with real exercise

# Chapter 5

# Deployment and evaluation

This chapter focuses on the deployment, release and feedback of the Genghis authoring tool, including the demonstration issued during the presentation to University Carlos III's professors, as well as issues the first test users reported.

## 5.1 Quality Assurance

After the redesign of the main view and with little programming left to do, an extensive quality assurance was made before the tool's deployment to the production server, where end users will finally be able to interact and use the tool.

This quality assurance process was made initially by assigning issues in https://github.com, where the source code was hosted. Github offers a series of tools for bug and feature issuing, including comments, images and actual code references (sections, files and even commits).

There were several alternatives to using github's issue system, such as trello or redmine. While trello was far too simple and maybe not too adequate for issues (as much as for doing To Do lists and more general development tasks), redmine offered an incredibly vast array of tools and option for issues and bugs, far too much for our purposes. These problems, together with the fact that the code was already on github, led the team to use github issues over all the other possibilities.

Quality assurance went as expected, some bugs arose, such as:

- Issues when editing variables, such as **not saving the new variable format** (integer or float): When editing a variable (that is, changing either the minimum, maximum or type), the type would be changed much like the other parameters but an unfortunate typo in the code left from a test run of the function, always set the type to integer on edition. The fix was simple enough: make the variable change according to the POST parameter on the request.

- **The preview pane would not display exercises correctly**: As one of the trickiest part of the code, the preview pane suffered many changes during its implementation. This bug in particular manifested at random times when a user wanted to see a preview pane and could not get the latest version of the exercise. The first solution was to simply refresh the site when this problem occurred, but that was far from a permanent solution. The team managed to narrow it down to two things: response speed was too slow, since it was run on a python simple HTTP server, a change to an Apache tomcat promptly solved this issue. Secondly, the cache was active and sometimes it would use the content already on the site. To solve this, cache was disabled when requesting the preview pane.
- **Design issues**: misplaced buttons and errors in responsive mode were some of the minor issues assigned.
- **Adding of feedback possibilities**: Rather an enhancement than a bug, there had to be a way for professors to give feedback on the tool. For this purpose, a feedback button was added.
- **A single save button**: Originally, each section of the exercise (title, question, solution and hints) had its own save button, but seeing as this could lead to confusion, all these buttons were unified under a single save button except for the hints, whose system was left intact, much like the variables (each is added and edited individually)
- **Floating point random numbers could be added without a step**: a small error in the controller and the database which was promptly changed in the fields in the database to be non-nullable and the controllers to always check the existence of at least one numeric character.
- **Fixing of   in parser**: One of the initial versions of the parser would incorrectly parse the non-breaking space and lead to an incorrect exercise markup.

Once this issues were corrected and the respective changes committed to the remote git repository, it was time to deploy the Genghis authoring tool in the production server.

## 5.2 Deployment

The Genghis tool was to be deployed in one of the university's servers, where a public IP will allow anyone from the Internet to access the tool (with the right credentials of course).

The deployment was carried swiftly by the University's IT department. Since Apache was already installed, a simple

```
git clone https://github.com/kadaki01/genghis.git
```

will suffice to get all the required files up and running. A more detailed explanation of the entire Git process is explained in Appendix A

There was also the subject on integrating with the GEL platform, the environment on which Genghis will run. This platform was also in charge of the authentication of users, as well as the

upload of videos under the corporate UC3M YouTube account, so all professors could send their videos to YouTube through the YouTube API.

The development of the GEL platform sent our team a set of session parameters from which the authoring tool could get the authenticated user from the university's LDAP. Although not ideal, the token was passed along in the session to the controllers from which the tool could store the author and display the exercises that user had already made. A simple matter of adapting the controller and views for the inclusion of the author was easy enough, since the model and the database were already prepared for this.

The final extra step that had to be done was to download the khan-exercise package from Khan Academy's Github page and run it under an Apache Tomcat or similar.

## 5.3 Presentation and evaluation

Once the deployment phase was over and the Genghis authoring tool was set up, the team set up a small group of professors for an initial testing phase, from which to gather important feedback and a small evaluation of the tool that could potentially allow us to discover unknown bugs or add features professors may need or think as an useful improvement to the interface or functionality.

This team of beta testers were a small group of professors from the physics department at UC3M, some of which took part in the initial brainstorm and followed the development and progress of the authoring tool as it evolved over time. The task was to create a certain number of real exercises to test the tool in a real life scenario.

For these tests, the team recommended the use of a text file per exercise, in which professors should write the areas of the exercise (Title, statement, solution and hints), as well as variable and a small test case. This way a more efficient approach was ensured as it would reduce problems arising from an exercise being malformed.

These professors were set to create a set of 27 exercises for the *Curso Cero* taking place next year. These exercises would be added to the physics course for students to take, making this the first test for the Genghis authoring tool. These exercises were based on high school level of physics, mostly on the area of linear motion and the basics of electricity. A small selection of these is presented below for the record and to show the form of exercises the Genghis authoring tool would have to be prepared to parse.

- **Inclined plane**: This exercise was meant as a small review of the basic concepts of rectilinear movement in a frictionless ramp, on which an object is moving up the ramp at an initial speed and the student is meant to find the distance before stopping, given the angle of the ramp. Regarding the authoring this exercise had to undergo, there was only one variable to present, the initial velocity the object was moving at, chosen to be integer numbers, ranging from 2 to 20 meters per second. The angle was set to be a constant of 20.

  This exercise also required the insertion of an image (Shown in figure 5.1) to the statement, in order to help the student picture the exercise setup. The picture module was ready for inserting any image from the internet, but not for hosting it, hence requiring the professor

to upload it to an external website and link the image to TinyMCE's image plugin.

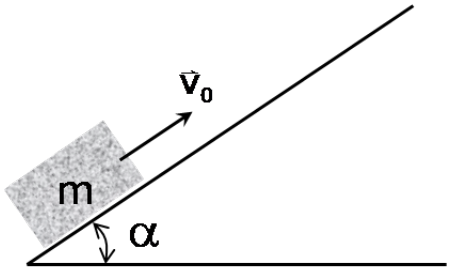


Figure 5.1: Inclined plane exercise image

- **Field strength**: Meant as a review of the basics of the fields different charges make on each other, this exercise had 4 variables: The charges of 3 particles and the distance between two of them. The rest of the distance were set as constants in the exercise statement. As it asked for the total strength on one of the charges, this particular exercise posed quite a challenge on the solution box, since the formula that correctly renders the solution is shown in formula 5.1.

$$\left| 90 * Q_1 * \left( \frac{Q_2}{(D_{12})^2} + \frac{Q_3}{(D_{12} + D_{23})^2} \right) \right| \tag{5.1}$$

  Although computationally simple to solve, this exercise's solution formula was both long and complex, including an absolute value, several variables, squares and the navigation of several nested parentheses.

- **Work and energy**: This 3 variable exercise was developed to test students on the movement of an object on a straight line of a certain mass and a stopping force acting on it, questioning the initial velocity if the object traveled a variable distance. The particular element of this exercise was not the solution (Shown in formula 5.2), but the hints, for there were three hints and one of them includes an image to be inserted.

$$v_0 = \sqrt{\frac{2Fd}{m}} \tag{5.2}$$

These were, as mentioned before, a small set of the 27 exercises parsed to Khan Academy styled markup using Genghis' authoring tool for the physics course.

Very valuable information and feedback was obtained from these exercises, both bugs and features, along with positive evaluations on certain aspects.

- **Link to the rest of GEL's platform**: It was requested to have a link to access the rest of the GEL platform without having to go back on the browser. A set of links marked as breadcrumbs were added for this purpose.

- **The area of the LaTeX formulas was poorly marked**: TinyMCE's LaTeX plugin was a very powerful tool but a bit complex to begin with. For this reason it was added to

the plugin configuration a red colored box to delimit these formulas, in order for professors to have a clear way of locating the limits of their LaTeX formulas from a simple glance.

- **Remove TinyMCE's original subscript and superscript**: Since the addition of the LaTeX plugin for TinyMCE, there were two possible ways to add superscripts and subscripts, via the default buttons in TinyMCE or LaTeX, which lead to confusion. Removing the buttons from all TinyMCE boxes solved the issue.
- **Add logarithm in base 10 and the absolute value in the solution box**: a request by some professors lead to the addition of these two operands to the available set.
- **Add ± sign to error tolerance**: A small typo in which no sign would be shown in the box in charge of setting the tolerance the engine should have for the answer students give, leading to confusion as one could think it was only the upper/lower bound or both added together.

These bugs were all promptly solved and the changes pushed to the production server.

Despite the missing features and bugs, the tool was very well received among the professors, praising the clean interface, simple usage and powerful tools available within TinyMCE's editor, specially the LaTeX editing plugin.

The very positive evaluation and the feedback obtained from these test exercises, together with the creation of a video tutorial to help incoming professors feel more conformable with the use of the tool, were the factors that pushed the decision to open the Genghis authoring tool to all other courses in the *Cursos cero*.

Professors from the chemistry and mathematics departments started working on their corresponding exercises for their courses, as well as some physics professors for some additional exercises added in the last days.

The exercises ranged in content type, length, solution complexity and number of variables but overall the experience was a complete success, with the following statistics:

- The exercises were for a total of 3 different courses.
- 35 professors creating and editing exercises.
- 135 exercises in total.
- 286 Variables, from which:
  - 211 were integers.
  - 75 variables as floats.
- 231 Hints.
- 48 exercises did not accept any form of error.
- 46 exercises did not have any rounding enabled for the final solution.
- 95 used the LaTeX editor for the solution.

Once finished, all the exercises were moved to the production Khan Academy server set up for the *Curso cero*, where students would be able to access the exercises and courses once signed up and the credentials were managed and sent to the corresponding user. This platform was also prepared to gather a series of analytics on the students behavioral patterns regarding the exercises, such as the number of hints used, time of the day the exercises are completed, time difference between exercise resolution and video completing levels, among many others. All this data resulted in very valuable information for future courses.

# Chapter 6

# Conclusions and Future work

This chapter serves the purpose to conclude the document, evaluating the project, the development process and the feedback, also discussing in some depth the future work that could be added to further enhance the tool and increase its usability.

## 6.1 Conclusions

The first and most important conclusion that can be drawn from this project is that Genghis and its authoring tool is a working web application that has helped professors make dozens of exercises at the University for the Massively online open courses taking place.

From the objectives in section 1.2 and based on the feedback from professors and their evaluation of the tool, it can be noted that the following were met and to what extent:

- **The tool must be simple**: Feedback from professors showed us that the tool was very simple to use and, except for some specific cases, little explanation or tutoring had to be done to instruct on its use.
- **The design must be clean and efficient**: The design update from the GEL team, together with the simple user interface and homogeneous look and feel in the overall GEL platform lead lead to a success in this particular objective.
- **The user experience as a whole has to be satisfactory**: No visual glitches made it to the production environment.
- **The process of creating an exercise must be straight forward**: The initial brainstorm and the change in CSS styles aided to create a simple exercise creation window that would follow the order of the actual exercise to be created.
- **Genghis must be finished in time for its final purpose: to be used by professors to create exercises for new, upcoming MOOCs**: The tool was finished in time, for the use of a beta-testing group and the final user base.

- **The tool must be integrated in the GEL platform, from which professors would access to their exercises and data**: Thanks to the GEL team, the integration was simple and straight forward, with professors being able to browse seamlessly from the Genghis authoring tool to the rest of the platform.
- **The tool must be hosted using version control to avoid code being lost and easing the deployment phase**: Git was used through the entire creation and development, simplifying the version management and deployment process.

The methodology used for version control and debug and release server were optimal and worked perfectly, the custom MVC pattern used served its purpose, although, as previously mentioned, the use of a framework will most likely help in the efficiency and avoid some code redundancy, specially with object mapping, front end engines and route resources, like the ones offered in the Laravel framework.

There were, however, some limitations to the Genghis authoring tool that could be added in future versions and greatly improve its versatility and usability, such as:

- The presented version of the tool could only make Fill-in-the-blank exercises.
- Exercises could only be managed by a single person at a time, meaning concurrent editing was not possible without race conditions.
- The LaTeX module turned to actual LaTeX during edition, leading to possible confusion for people not familiar with the language.
- Limited number of operations in the solution box.

These are, however, taken into account and further discussed with their solutions in section 6.2.

Another aspect worth mentioning is the interest several parties took in the tool and its development. A Dutch software company approached the Genghis development team because of similar tool was being built at their labs, and Genghis could further improve their advances as well as offer a new array of possibilities. The team was also contacted by a software developer in a university in South America with a great interest in deploying the tool in their servers for private use within their teaching staff.

Finally, the creation of two publications, an article and a chapter for a book, both cited below, prove the interest the community has in learning more about MOOCs and the experience UC3M has had with not only the Khan Academy MOOC, but with Genghis and its authoring tool as well.

The first publication was the article accepted for the IEEE Global Engineering Education Conference, EDUCON 2014, Istanbul, Turkey, 2014.

> Delgado Kloos, C. ; Munoz-Merino, P.J. ; Munoz-Organero, M. ; Alario-Hoyos, C. ; Perez-Sanagustin, M. ; Parada G, H.A. ; Ruiperez, J.A. ; Sanz, J.L. (2014) "*Experiences of Running MOOCs and SPOCs at UC3M*". IEEE

Followed by the chapter for the book "*Furthering Higher Education Possibilities through Massive Open Online Courses*"

> Munoz-Merino, P.J. ; Ruiperez, J.A. ; Sanz, J.L.; Delgado Kloos, C. (2014)"*Assessment Activities in Massive Open On-line Courses: Assessment Activities in MOOCs*". IGI

## 6.2 Future work

Although the Genghis experience at UC3M is concluded and professors have a working tool to develop their exercises, the idea had such a great potential that was a rework was programmed for the late summer of 2014.

The Genghis project would now branch into what is now known as Open Genghis, a tool with the same purpose as the original Genghis tool, but with many improvements built from scratch, starting with the architecture.

Open Genghis is based on Laravel 4.1, the (at the time of writing) most up to date version of the Laravel framework, and greatly improves the custom MVC by using the following tools and practices:

**Improvements on the limitations**

As mentioned in section 6.1, the tool had four main limitations and the effort in overcoming such problems could result in a more versatile tool. These changes could be approached as follows:

- **The presented version of the tool could only make Fill-in-the-blank exercises**: Khan Academy markup allows for a greater number of exercise formats, including True/False, multiple choice, single choice and graph-based solutions. The extensive documentation on these exercise types would make the adding of these exercise types very simple. This could be made by:
  - Creating a new view or partial view for each exercise type.
  - Adapting the models for these new exercise types.
  - Adapting of the controllers, possibly by creating a base controller from which all other exercise controllers could inherit.
- **Exercises could only be managed by a single person at a time, meaning concurrent editing was not possible without race conditions**: This could prove hard to do without some heavy load on the server and the database, but using libraries such as EtherPad or ShareJS could make this possible.
- **The LaTeX module turned to actual LaTeX during edition, leading to possible confusion for people not familiar with the language**: Further development on the editor could allow a user to optionally create formulas in a more "drag-and-drop" style, like Microsoft Word's formula editor.

- **Limited number of operations in the solution box**: The plugin was left modular so the addition of more formulas and operands could be done easily, either by request or including a broader array of possibilities by default.

### ORM

Genghis' authoring tool had a basic form of object relationship mapping, but the use of Laravel meant the custom implementation was no longer necessary. A complete code sample of the models is provided in Appendix B in page 73.

### Authentication

Laravel offers a wonderful authentication plugin built in with the framework, a simple matter of changing parameters in *app/config/auth.php* detailing the form of authentication and storage procedure would lead to a much simpler code when checking for authenticated users, as demonstrated below, a one line way of checking whether the user is authenticated by using one of Laravel's static classes instead of a more tedious, manual way.

```
if (Auth::check())
{
    // The user is logged in!
}
```

### Resources

Laravel offers routing resources to help simplify the task of routing. Resources are simple groups of method put together under a single route that must lead to a controller with at least the following methods:

| Verb | Path | Action | Actual route name |
|---|---|---|---|
| GET | /resource | index | resource.index |
| GET | /resource/create | create | resource.create |
| POST | /resource | store | resource.store |
| GET | /resource/{resource} | show | resource.show |
| GET | /resource/{resource}/edit | edit | resource.edit |
| PUT/PATCH | /resource/{resource} | update | resource.update |
| DELETE | /resource/{resource} | destroy | resource.destroy |

Table 6.1: Resource controller actions

These 7 methods, ideal for a Genghis exercise and the actions they undergo (listing, creation, saving, showing, editing, updating and deleting) are 7 routes that do not need to be specified in the *app/routes.php* file, but only using the next line will suffice.

```
Route::resource('exercises', 'ExercisesController');
```

### Laravel blades

The use of a master blade to keep the frame, also known as the part of the view that is the same through the entire project, was a very important improvement to make, as code repetition became minimal.

The use of a templating engine was also proven very useful for code cleaning and effortlessly navigating data structures in the view.

**Bundling and minification**

Another great improvement would be to minify (obfuscating) and bundle together all CSS files and JavaScript respectively. This would greatly reduce the number of requests to the view and drastically improve loading times, specially with JavaScripts blocking problem.

**Security**

One key feature not yet contemplated is the increase in security. Forcing the use of SSL through HTTPS only connections would reduce the risk of having someone sniff the local network and easily see the traffic.

It would also be a very wise measure to add an Anti-Forgery token to all forms (including AJAX requests) to avoid Cross Site Request Forgery attacks.

# Appendix A

# Git deployment

Git is a very versatile tool that not only serves as a source control software, but can also be used with other tools for code reviewing and manual accepting of merges with Gerrit and Jenkins. If used correctly, it can also be a very helpful tool when deploying to production from a master branch, the complete process is explained below:

1. Create an empty git repository

   ```
   $ git init
   ```

2. Link to a remote server

   ```
   $ git remote add master https://myremote.address.com/repository
   ```

3. Add files and folders

   ```
   $ git add myFile.php
   $ git add myFolder/
   $ git commit -m "First commit, addition of files"
   $ git push origin master
   ```

4. Change and update these files

5. On the deployment server, simply clone the repository.

   ```
   $ git clone https://myremote.address.com/repository
   ```

6. When updating to a newer version form the master branch, simply pull any changes.

   ```
   $ git pull origin master
   ```

# Appendix B

# Model mapping

Mapping the models using Laravel is a very simple task. The framework already tries to get the model from a table in the database that shares a name with model, but leaving an option to be specified. In our case, the three models become:

## B.1 The variable model

```
<?php

class Variable extends Eloquent {

protected $table = 'variables';

protected $guarded = array();

public static $rules = array();

}
```

## B.2 The exercise model

The exercise model has to specify the one-to-many relationship with the variable table (as one exercise can have several variables).

```
<?php

class Exercise extends Eloquent {
```

```
protected $table = 'exercises';

protected $guarded = array();

public static $rules = array();

public function variables()
    {
        return $this->hasMany('Variable', 'exercise_id');
    }
}
```

## B.3 The user model

```
<?php

use Illuminate\Auth\UserInterface;
use Illuminate\Auth\Reminders\RemindableInterface;

class User extends Eloquent implements UserInterface, RemindableInterface {

protected $table = 'users';

protected $hidden = array('password');

public function getAuthIdentifier()
{
    return $this->getKey();
}

public function getAuthPassword()
{
    return $this->password;
}

public function getReminderEmail()
{
    return $this->email;
}

}
```

# Bibliography


[1] P. Van Rosmalen, H. Vogten, R. Van Es, H. Passier, P. Poelmans, and R. Koper. Authoring a full life cycle model in standards-based adaptive e-learning. *Educational Technology and Society*, 2006.

[2] B. Soo Ong and A. Grigoryan. Moocs and universities: Competitors or partners? *International Journal of Information and Education Technology*, 2014.

[3] Spain internet usage, population and telecommunications report. `http://www.internetworldstats.com/eu/es.htm`.

[4] Youtube. `https://www.youtube.com/`.

[5] Khan academy. `https://www.khanacademy.org/`.

[6] Khan/khan-exercises repository at github.com. `https://github.com/khan/khan-exercises`.

[7] Cursos cero — uc3m. `http://uc3m.es/ss/Satellite/UC3MInstitucional/en/TextoMixta/1371206785343/Cursos_cero`.

[8] About the team — khan academy. `https://www.khanacademy.org/about/the-team`.

[9] Interview with salman khan — new york times. `http://www.nytimes.com/2014/01/28/science/salman-khan-turned-family-tutoring-into-khan-academy.html?_r=0`.

[10] Coursera's official site. `https://www.coursera.org/`.

[11] edx official site. `https://www.edx.org/`.

[12] The tech — mit blog. `http://tech.mit.edu/V134/N39/highschooledx.html`.

[13] Persistence (computer science). `http://en.wikipedia.org/wiki/Persistence_(computer_science)`.

[14] Nosql. `http://nosql-database.org/`.

[15] Mvc process. `http://commons.wikimedia.org/wiki/File:MVC-Process.svg`.

[16] Ruby on rails' official site. `http://rubyonrails.org/`.

[17] Laravel's official site. `http://laravel.com/`.

[18] jquery's official site. `http://jquery.com/`.

[19] jquery ui's official site. http://jqueryui.com/.

[20] Twitter bootstrap's official site. http://getbootstrap.com/.

[21] Git's official site. http://git-scm.com/.

[22] Analyzing critical rendering path performance web fundamentals. https://developers.google.com/web/fundamentals/performance/critical-rendering-path/analyzing-crp.

[23] Tinymce. http://www.tinymce.com/.

[24] Scorm's official website. http://scorm.com/scorm-explained/.

[25] Ims qti implementation guide. http://www.imsglobal.org/question/qtiv2p1/imsqti_implv2p1.html.